UNISYS
imagine it. done.

Unisys ClearPath Enterprise Servers

Relational Database Fast Load User Guide

ClearPath OS 2200 Release 12.0

June 2009 7847 4889–007

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THIS DOCUMENT. Any product or related information described herein is only furnished pursuant and subject to the terms and conditions of a duly executed agreement to purchase or lease equipment or to license software. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, special, or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Notice to U.S. Government End Users: This is commercial computer software or hardware documentation developed at private expense. Use, reproduction, or disclosure by the Government is subject to the terms of Unisys standard commercial license for the products, and where applicable, the restricted/limited rights provisions of the contract data rights clauses.

Unisys and ClearPath are registered trademarks of Unisys Corporation in the United States and other countries.
All other brands and products referenced in this document are acknowledged to be the trademarks or registered trademarks of their respective holders.

Unisys ClearPath Enterprise Servers

Relational Database Fast Load

User Guide

ClearPath OS 2200 Release 12.0

7847 4889–007

Unisys ClearPath Enterprise Servers

Relational Database Fast Load

User Guide

ClearPath OS 2200 Release 12.0

7847 4889–007

Bend here, peel upwards and apply to spine.

Contents

Section 1. Introduction

1.1. Documentation Updates 1–1
1.2. Overview 1–2
1.2.1. Description of XRLOAD 1–2
1.2.2. Advantages of Offline Loading Over Online Updating 1–2
1.2.3. Comparison of XRLOAD with the RDMUTL Processor 1–2
1.2.4. Summary of XRLOAD Advantages and Benefits 1–3
1.3. Interacting with XRLOAD 1–4

Section 2. Site Requirements and Considerations

2.1. System Environmental Requirements 2–1
2.1.1. Installation 2–1
2.1.2. UDS Files 2–2
2.2. XRLOAD Processor Requirements 2–3
2.2.1. Considerations for Tables 2–3
2.2.2. Considerations for Storage Areas 2–4
2.3. Recovery Rules 2–5
2.3.1. Considerations Regarding IRU Recovery 2–5
2.4. Recovering from an Incomplete Load 2–6
2.4.1. Disabling Corrupted Storage Areas 2–6
2.4.2. Recovery Procedure 2–6

Section 3. Preparing Input Files and Text

3.1. File Type Requirements 3–1
3.1.1. Disk Files 3–1
3.1.2. Tape Files 3–2
3.2. Input Text Requirements 3–2
3.2.1. Rules for Preparing External, Internal, and User Files 3–2
3.2.2. Guidelines for Preparing NCHARACTER Data 3–3
3.3. EXTERNAL Format Files and Data Rules 3–3
3.3.1. Definitions 3–3
3.3.2. Rules for Numeric Data 3–3
3.3.3. Rules for Character Data 3–4
3.3.4. Rules for Datetime Data 3–5
3.3.5. Rules for BLOB Data 3–5
3.4. INTERNAL Format Files 3–6
3.4.1. Input File Requirements 3–7

3.4.2. UCS COBOL Data Types 3–7
3.4.3. REAL, DOUBLE PRECISION, and FLOAT Data Types 3–8
3.4.4. DATE Data Type Assignment 3–9
3.4.5. TIME Data Type Assignment 3–9
3.4.6. TIMESTAMP Data Type Assignment 3–9
3.4.7. BLOB Data Type 3–10
3.4.8. Null Indicators 3–10
3.5. USER Format Files 3–12
3.5.1. Null Indicators 3–12

Section 4. Loading the Data

4.1. Calling XRLOAD 4–1
4.1.1. Runstream Model 4–1
4.1.2. Notation Conventions 4–2
4.1.3. XRLOAD Calling Runstream Lines 4–2
4.2. Using the LOAD Command 4–4
4.2.1. Syntax 4–4
4.2.2. Notation Conventions 4–4
4.2.3. LOAD Command Parts 4–5
4.2.4. FILE Part Considerations for Multiple Inputs 4–7
4.2.5. Format of Item-Position Field 4–7
4.2.6. Item-Position Parts 4–8
4.2.7. Format of NULLIF Clause 4–8
4.2.8. NULLIF Clause Parts 4–9

Section 5. The Process

5.1. How XRLOAD Loads Data 5–1
5.1.1. The XRLOAD Process 5–1
5.1.2. JETLOAD Output 5–2
5.1.3. JETLOAD Completion Messages 5–2
5.2. Improving Load Performance 5–3
5.2.1. Table Description 5–3
5.2.2. Input File 5–3
5.2.3. Execution of XRLOAD 5–4

Appendix A. UCS COBOL Data Definitions

A.1. CHARACTER and NCHARACTER Data A–1
A.2. DECIMAL, INTEGER, SMALLINT, and NUMERIC Data A–2
A.2.1. DECIMAL Data A–2
A.2.2. INTEGER Data A–2
A.2.3. SMALLINT Data A–3
A.2.4. NUMERIC Data A–3
A.3. REAL, DOUBLE PRECISION, and FLOAT Data A–4
A.4. DATE, TIME, and TIMESTAMP Data A–4
A.4.1. Comparable UCS and RDMS Datetime Data Types A–4

A.4.2. DATE Data A–4
A.4.3. TIME Data A–5
A.4.4. TIMESTAMP Data A–6
A.5. BLOB Data A–6

Appendix B. Sample Runstreams

B.1. Table Definition B–1
B.2. XRLOAD Execution Runstream B–2
B.3. Runstream Generated by XRLOAD B–2
B.4. Execution Result of XRLOAD B–3

Appendix C. Relational Database Fast Load with Empty Input File

C.1. The Process C–1
C.2. Table Def C–1
C.3. XRLOAD Execution Runstream C–2
C.4. Runstream Generated by XRLOAD C–2
C.5. Execution Result of XRLOAD C–3

Appendix D. Loading into Partitioned Tables

D.1. The Process D–1
D.2. PTLOAD Output D–2
D.3. PTLOAD Completion Messages D–3
D.4. Special Requirements for Partitioned Tables D–3
D.5. Improving Load Performance D–3
D.6. Loading into a Partitioned Table – Once D–4
D.7. Loading into a Partitioned Table – Parallel D–5
D.7.1. Loading into Each Index in Parallel D–6
D.7.2. Loading into Each Partition in Parallel D–7
D.8. Table Definition D–8
D.9. XRLOAD Execution Runstream D–8
D.10. Runstream Generated by XRLOAD D–10
D.11. Execution Result of XRLOAD D–11

Appendix E. Relational Database Fast Load Error Messages

E.1. Error Message Description E–1
E.2. Error Categories E–4
E.3. Initial Errors E–6
E.4. Syntax Errors E–7
E.5. Errors 30000 to 99999 ..... E–11
E.6. Definition Errors E–11
E.7. Generation Errors E–14
E.8. Logical Errors E–15
E.9. Facility Errors (ER CSF$ and ER FACIL$ Errors) E–18
E.10. I/O Errors E–18

E.11. Contingency Error E–20
E.12. Fatal Errors (JETLOAD or PTLOAD Internal Errors)..... E–20

Index 1

Figures

1–1. The XRLOAD Loading Process 1–5

D–1. Loading Process into a Partitioned Table – Once D–4
D–2. Loading Process into Each Index in Parallel D–6
D–3. Loading Process into Each Partition in Parallel D–7

Section 1
Introduction

This guide explains how to load data into Relational Data Management System (RDMS) databases with the XRLOAD processor. It illustrates the XRLOAD process from beginning to end and contains sample runstreams.

This guide is for database administrators, who generally supervise RDMS operations. However, anyone who is responsible for loading data into RDMS databases can use this guide.

This guide contains information that was available at the time of publication. Technical changes not available at the time of publication are supplied in PLE 18303116. You can obtain a copy of this PLE by accessing the following Unisys Support Online (USO) Web page:

```
http://www.service.unisys.com
```

This section describes the advantages of offline loading over online updating, compares XRLOAD to the RDMUTL processor LOAD command, summarizes the advantages and benefits of XRLOAD over RDMUTL, and briefly describes how to interact with XRLOAD.

1.1. Documentation Updates

This document contains all the information that was available at the time of publication. Changes identified after release of this document are included in problem list entry (PLE) 18595460. To obtain a copy of the PLE, contact your Unisys representative or access the current PLE from the Unisys Product Support Web site:

http://www.support.unisys.com/all/ple/18595460

Note: *If you are not logged into the Product Support site, you will be asked to do so.*

1.2. Overview

1.2.1. Description of XRLOAD

The XRLOAD utility processor implements fast initial offline loading of a large amount of data into an RDMS database. XRLOAD is a stand-alone utility. It does not access UDS Control or RDMS.

1.2.2. Advantages of Offline Loading Over Online Updating

Offline loading offers two primary advantages over online updating:

- It minimizes processing time.
- It prevents degradation of system throughput, as is the case with online updating when data cannot be accessed for a period of time.

Presently, whenever a large amount of data comes from outside the system and the data needs to be updated, updating on the system with RDMS creates a long audit trail, disturbs caches, and degrades the hit rate.

1.2.3. Comparison of XRLOAD with the RDMUTL Processor

XRLOAD includes its own storage manager, JETLOAD and PTLOAD, to build RDMS B-trees.

XRLOAD uses the SORT processor to sort records for loading

- Into secondary index and unique constraint B-trees
- From a file that is not sorted in the table's primary key or specified index’s key order

XRLOAD loads data faster than the RDMUTL processor's LOAD command and is noticeably more effective if the database contains many secondary indexes and unique constraints. JETLOAD always reads presorted record sets during the loading process; it does not insert data randomly into secondary index and unique constraint B-trees.

XRLOAD processes one LOAD command per call, while RDMUTL allows multiple LOAD commands per call. XRLOAD processes one table per LOAD command, while RDMUTL allows multiple tables per LOAD command.

XRLOAD allows multiple input files per LOAD command, while RDMUTL allows only one input file per LOAD command. XRLOAD merges and sorts the input files during the load process.

The LOAD commands for the XRLOAD and RDMUTL processors are syntactically similar.

Unlike the RDMUTL processor LOAD command, XRLOAD does not provide the following clauses:

- APPEND/RESUME

  Loads data into a table piecemeal (that is, the table does not have to be empty).
- START RECORD

  Specifies the line at which loading is to begin.
- STOP RECORD

  Specifies the line at which loading is to end.
- IGNORE REFERENCES

  If omitted, verifies foreign key relationships for the table.
- RESTARTABLE

  YES specified on this clause makes the load operation automatically restartable.
- COMMIT EVERY

  Specifies the frequency at which to commit the loaded records.

INDEX clause is provided only for partitioned tables.

1.2.4. Summary of XRLOAD Advantages and Benefits

Compared to RDMUTL, XRLOAD has two main advantages:

- It takes less time to load a large amount of data.
- It has less impact on system resources.

Consider also the following benefits of XRLOAD:

- Construct a very large database in a short period of time.

  Saving time and performance degradation when loading data while the system is running is not a concern. This results in a dramatic reduction in I/Os and elapsed time.
- XRLOAD does not use RDMS, but writes data directly onto disks. Thus, it does not disturb the UDS cache hit ratio.

  No auditing is performed and I/O size is not limited by RDMS schema definitions.
- XRLOAD performs bulk I/O, thus reducing the number of I/Os, and it does not disturb the Extended Processing Complex (XPC) cache hit ratio.
- XRLOAD loads data directly without RDMS and performs asynchronous large I/Os. Thus, elapsed time is determined by CPU or I/O time, whichever is longer.
- Compatible syntax and RDMUTL INTERNAL format input data allows for an easy transition from RDMUTL to XRLOAD.

- Use output data in INTERNAL format from the RDMS UNLOAD statement as input to XRLOAD. This reorganizes an RDMS database using the RDMS UNLOAD statement and XRLOAD.
- XRLOAD loads multiple input files into a table without a merge process, using COBOL or other steps when there are input files.
- If a system error occurs, XRLOAD can be rerun from the beginning. A recovery point is not an issue.
- XRLOAD applies the load factor in the following cases; RDMUTL does not:
  - Into secondary index and unique constraint B-trees from a LOAD command that has no INDEX clause (RDMUTL requires the INDEX clause.)
  - From a file that is not sorted in the table's primary key order
  - Into a table that contains no primary key constraints

1.3. Interacting with XRLOAD

Briefly, interacting with XRLOAD is summarized in the following steps:

1. Prepare a file that contains source data for loading. This is called the input file.
2. To prepare the file, use any file or text editor, or COBOL or any other language program.
3. Call XRLOAD and enter a LOAD command.
4. XRLOAD obtains the information it needs from UDS system files and generates a runstream for the load. The generated runstream is added automatically.
5. From the runstream, XRLOAD calls JETLOAD or PTLOAD to perform fast initial loading into each individual B-tree. JETLOAD also projects items and builds a temporary system data format (SDF) file. It then calls the SORT processor to sort the records with projected items for each secondary index and unique constraint load.

Section 5 gives a more detailed description of the XRLOAD process. Figure 1–1 illustrates this process.

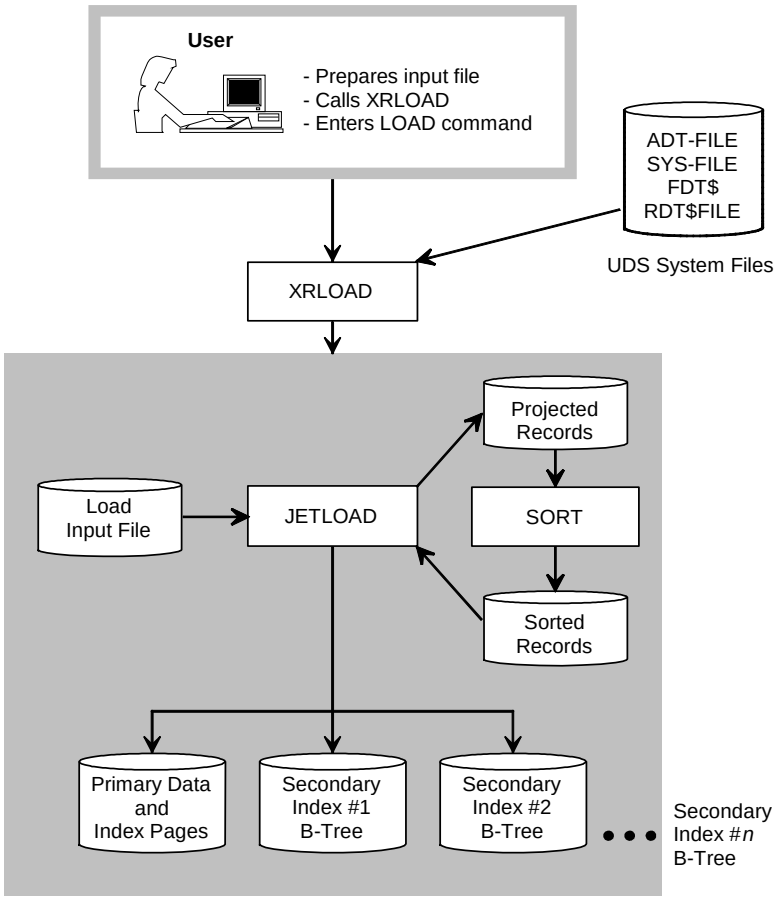


Figure 1–1. The XRLOAD Loading Process

Section 2
Site Requirements and Considerations

This section describes the environmental requirements for running XRLOAD. It also discusses points to consider before loading the data into tables, as well as storage area and recovery considerations.

2.1. System Environmental Requirements

Before attempting to run XRLOAD, prepare the environment by insuring that your site meets the following requirements:

- RDMS software level 12R1 or higher must be installed and available on your system.
- Mass storage capacity must be of sufficient size to run XRLOAD. Approximately five times the size of the input file is needed.

2.1.1. Installation

XRLOAD is installed using SOLAR in mode A.

If XRLOAD is not yet installed at your site, refer to the following two manuals:

- *Software Products Installation Guide*
- *SOLAR User Reference Manual*

2.1.2. UDS Files

When loading into partitioned tables, UDS files FDT$, RDT$FILE, and PART$FILE, of the application group being referred to, must be available to XRLOAD for the duration of its runstream-generating process.

File RDT$FILE and PART$FILE are usually defined as an internal storage area with DOMAIN UDS, which UDS assigns exclusively while the application group is active. Therefore, change the DOMAIN attribute of file RDT$FILE and PART$FILE to USER-UDS, as in the following example.

Changing the DOMAIN attribute of file RDT$FILE and PART$FILE to USER-UDS opens files FDT$ and RDT$FILE and PART$FILE to all users. These files are not as secure, therefore, during the XRLOAD process.

Example

```
@dd,e ,,application-group-name
update storage-area rdt$file version uds for schema tcs.
remove domain.
add domain user-uds.
process storage-area rdt$file version uds for schema tcs update.
process storage-area rdt$file version uds for schema tcs report.
exit.
```

When multiple loadings with the changed DOMAIN attribute of UDS are executed in parallel, the DOMAIN attribute might change again in another run during the runstream-generating process of XRLOAD. If this situation occurs, XRLOAD might produce unpredictable errors.

2.2. XRLOAD Processor Requirements

Before calling XRLOAD, certain points must be considered about the tables and storage areas that the data will be loaded into.

2.2.1. Considerations for Tables

Tables are subject to the following requirements and restrictions:

Table Element	Requirements and Restrictions
Tables	The table • Must have been created by the RDMS CREATE TABLE statement before XRLOAD is called to load data into it. • Cannot include a column definition with a nondefault character set or collation. • Cannot include IDENTITY column or GENERATED expression. • Must be available to XRLOAD.
Partitions	When loading into a partitioned table, change the state of the partition in which data is actually loaded to detached or hidden by the RDMUTL DETACH PARTITION or HIDE PARTITION command before calling XRLOAD.
BLOBs	XRLOAD does not load BLOB text itself. XRLOAD can load • LOB identifier unloaded by the RDMS UNLOAD statement with WITHOUT BLOB VALUES clause into a BLOB column. • NULL value into a BLOB column by specifying NULLIF clause (EXTERNAL format) or NULL indicator (INTERNAL/USER format).
XRLOAD processor	XRLOAD does not • Load data into views, but only into base tables. • Load data into a particular index in the table that is not partitioned. • Load global temporary tables. • Verify whether or not the user has INSERT privileges on the table. • Enforce referential (or foreign key) constraints and check constraints. If such constraints exist, XRLOAD ignores them.

Primary keys, secondary indexes, and unique constraints	These restrictions apply: • Primary keys, secondary indexes, and unique constraints cannot contain approximate numeric data type columns. • Primary keys cannot contain NULL ALLOWED columns if an INTERNAL or a USER format input file is used that is not sorted in the table's primary key order. • In case of loading into a particular index or partitions in a partitioned table, index keys cannot contain NULL ALLOWED columns if an INTERNAL or a USER format input file is used that is not sorted in that key order.
Default clause	XRLOAD does not apply a default value according to the default clause on the table definition. An input record on the input file must have a value for every column.

2.2.2. Considerations for Storage Areas

Storage areas that data will be loaded into are subject to the following requirements and restrictions:

Storage Areas	Requirements and Restrictions
Storage areas	The storage area must have been created by the UREP DD processor command CREATE STORAGE-AREA. The storage area must be registered and initialized by the UREP DD processor command PROCESS STORAGE-AREA...INSTALL. If the storage area is recataloged by the @DELETE and @CAT/@ASG,CP statements after being initialized, the storage area must be reregistered and initialized by the UREP DD processor commands PROCESS STORAGE-AREA ... DELETE and PROCESS STORAGE-AREA ... INSTALL. ***Note:*** *It is unnecessary to create and install implicit storage areas because RDMS and UREP perform this task automatically.* To deactivate the storage area before calling XRLOAD, the IRU DOWN command must be used. Page size must be more than 112 words. It is acceptable, in the case of loading into a partitioned table, to DOWN only the storage area in which data is actually loaded. If the table's storage area is FILE-TYPE EXEC, XRLOAD assigns the underlying file exclusively for the duration of B-tree construction and frees it internally. If the EXEC storage area has a read or write key, assign the file with the read or write key before executing XRLOAD and free the file after XRLOAD terminates.

XRLOAD processor	XRLOAD does not support storage areas associated with more than one table definition. ***Note:*** *The use of multifile relations and storage area versions (where one table definition is associated with multiple storage area definitions) is allowed.* XRLOAD does not enforce CHECKSUM; therefore, the declaration of a table's storage area with CHECKSUM YES has no effect. XRLOAD does not support encrypted files. The table's storage area is initialized during the loading process, and all existing data is overwritten.

Caution

XRLOAD issues no error or warning messages in regard to losing existing data.

2.3. Recovery Rules

XRLOAD runs outside of the UDS architecture and therefore does not utilize integrated recovery.

2.3.1. Considerations Regarding IRU Recovery

Consider the following points about recovery before using XRLOAD:

- Ensure that no incomplete database updates exist that require IRU short or medium recovery in the data to be loaded.

Caution

To prevent database corruption, do not run XRLOAD on a database with valid updates that are not yet committed and then perform an IRU recovery procedure. XRLOAD has no mechanism to prevent this from occurring.

- Once XRLOAD is executed on a database, an IRU long recovery procedure using audit information that was taken before the XRLOAD process cannot be performed.

Caution

To prevent database corruption, do not perform an IRU long recovery procedure using audit information that was taken before the XRLOAD process began. XRLOAD has no mechanism to prevent this from occurring.

Unisys recommends that you perform an IRU static dump of the database. This will enable a long recovery procedure to be accomplished in the future.

For IRU recovery procedures, see the *Integrated Recovery Utility Operations Guide.*

2.4. Recovering from an Incomplete Load

When an error is detected, XRLOAD and its storage manager JETLOAD terminates, aborts the loading process, and issues error messages. If such a failure occurs, the incomplete database may not be accessible by general SQL commands and may produce unpredictable results.

As a precautionary measure, produce a dump of the database to be used for a long recovery procedure in the future. Use the IRU STATIC DUMP command or the RDMS UNLOAD statement.

If anything fails while using XRLOAD to load data, take immediate action:

1. Disable the corrupted storage areas.
2. Restore the database to its original state.
3. Reinitialize the table's storage area or areas.

2.4.1. Disabling Corrupted Storage Areas

If an error message is received during the XRLOAD loading process, use the IRU DOWN IMMEDIATE command to disable the corrupted storage areas.

To avoid further complications, do not wait for other threads that are accessing the storage areas in question to finish processing data.

2.4.2. Recovery Procedure

Take one of the following actions to recover from an XRLOAD failure:

- Call XRLOAD again and reenter a LOAD command.
- Restore the database to its original state using the dump from before the XRLOAD process was started.
- Empty the database by initializing the table's storage area or areas using UREP DD processor commands:
  - Execute the PROCESS STORAGE-AREA . . . DELETE command for the storage area. If the table uses multifile relations, execute the command on the storage area that contains the table's primary key.
  - Execute the PROCESS STORAGE-AREA . . . INSTALL command for the storage area. If the table uses multifile relations, execute the command for each storage area that contains a version of the table.

Note: *If XRLOAD is to be called again after initializing the table's storage area or areas using UREP DD processor commands, use the IRU DOWN command again to deactivate the storage area before calling XRLOAD.*

For more information about IRU recovery procedures, see the *Integrated Recovery Utility Operations Guide.*

For instruction on using the RDMS UNLOAD statement, see the *Relational Database Server for ClearPath OS 2200 SQL Programming Reference Manual.*

For details about UREP DD processor commands, see the *Repository for ClearPath OS 2200 Programming Reference Manual*.

Section 3
Preparing Input Files and Text

This section describes the requirements and restrictions for input files. It specifically addresses

- The file types allowed
- The characteristics of input text
- EXTERNAL, INTERNAL, and USER format input files
- Data types

It includes examples of

- A table definition
- Input files in EXTERNAL, INTERNAL, and USER format
- The loading of the input files into the table

3.1. File Type Requirements

The XRLOAD processor handles SDF files as input for database loading. Input files can reside on disk or tape.

3.1.1. Disk Files

Input files assigned to disk must meet one of the following format requirements:

- Files unloaded by the RDMS UNLOAD statement, in either EXTERNAL or INTERNAL format
- Files in COBOL Processor Common Input/Output System (PCIOS) format assigned in COBOL as

```
SELECT. . .ASSIGNED TO DISC
```

- Symbiont files assigned in COBOL as

```
SELECT. . .ASSIGNED TO PRINTER
```

- Files edited by other text editors

- Files processed by the file transfer protocol (FTP) from a personal computer
- Files exported from MAPPER REPORT by the MAPPER ELT function

3.1.2. Tape Files

Input files assigned to tape must meet one of the following format requirements:

- Files unloaded by the RDMS UNLOAD statement, in either EXTERNAL or INTERNAL format
- Files in COBOL PCIOS format assigned in COBOL as

```
SELECT. . .ASSIGNED TO DISC
```

Note: *Files in COBOL PCIOS format assigned in COBOL as follows cannot be used:*

```
SELECT. . .ASSIGNED TO TAPE
```

The following guidelines apply to tape files:

- The block size of a tape file cannot exceed 17,920 words.
- XRLOAD does not handle multifile reels.
- XRLOAD does not check tape file labels.

3.2. Input Text Requirements

When actually loading data with the LOAD command (see 4.2), indicate whether the file is in EXTERNAL, INTERNAL, or USER format.

3.2.1. Rules for Preparing External, Internal, and User Files

The following guidelines apply for EXTERNAL, INTERNAL, and USER format input files:

- If the input file contains descriptive text, XRLOAD ignores it. For more information about descriptive text, see the *Relational Database Server for ClearPath OS 2200 Administration Guide.*
- CHARACTER columns cannot contain Fieldata code.
- The size of a record in an input file cannot exceed 2,047 words.
- Records in an input file must have the same fixed length.

For guidelines on NCHARACTER data, see the next subsection.

3.2.2. Guidelines for Preparing NCHARACTER Data

NCHARACTER data cannot include shift codes:

- The UCS COBOL processor does not include shift codes in input data for items defined as USAGE IS DISPLAY-2.
- However, shift codes are included in input data created with some processors. For example, shift codes are included in input data created with the Interactive Processing Facility (IPF).

If your input file contains shift codes, remove them before using the file.

Example

Use the following example to remove shift codes:

```
@ed,du sdf-file-name.
exch ! 223
tcchar %
c /!%// all
exit
```

3.3. EXTERNAL Format Files and Data Rules

3.3.1. Definitions

- EXTERNAL format input file acceptable to XRLOAD consists of pure ASCII text data.
- Numeric values are represented in ASCII numeric expressions in input files. XRLOAD automatically converts ASCII numeric expressions to the appropriate format based on the corresponding column definition.

  Note: *XRLOAD does not support approximate numeric values (that is, REAL, DOUBLE PRECISION, and FLOAT) in EXTERNAL format input files.*

3.3.2. Rules for Numeric Data

Except for a pattern that represents a null value (that is, where the pattern matches the pattern specified in the corresponding NULLIF clause), each numeric value in an input file must meet all the following criteria:

- Values can include the characters +, -, 0 through 9, and a decimal point, with or without leading ASCII blanks.
- Values must be right-justified appropriately.
- Signs (+ and -) must be of SIGN LEADING SEPARATE form if specified in an exact numeric expression; that is, the sign must appear in column 1 followed by the zeros (for example, +0000). If a sign is omitted, XRLOAD assumes the value to be a positive number.

- Decimal points must be specified as an ASCII "." (056_8).
- -0 is loaded as +0.

The following rules also apply:

- If the LOAD command omits the FORMAT clause, XRLOAD uses the corresponding column definition, as follows. XRLOAD also assumes that each value is delimited by a single ASCII blank.

DECIMAL and NUMERIC	Precision+1 If the column definition specifies scale, XRLOAD assumes the decimal point to be connected with the input value. Precision+1+1, therefore, is the number of characters assumed by XRLOAD for columns with scale.
INTEGER	12 characters
SMALLINT	7 characters
REAL	14 characters
DOUBLE PRECISION	25 characters
FLOAT	If precision is greater than 27, then 25 characters; otherwise, 14 characters.

- Specify characters longer than the corresponding column definition by using the FORMAT clause (see 4.2).

  The maximum number of characters allowed for each exact numeric value is 23 consecutive characters with or without the decimal point. The maximum number of characters allowed for each approximate numeric value (REAL, DOUBLE PRECISION, and FLOAT) is 24. For both exact and approximate values, XRLOAD automatically checks to determine whether the actual value can be stored in the column. If an overflow occurs, XRLOAD returns an error.

3.3.3. Rules for Character Data

Character data must meet the following criteria:

- The value must be left-justified.
- The pattern cannot be longer than the corresponding column definition.
- For kanji data, shift codes cannot be stored in the file.

3.3.4. Rules for Datetime Data

EXTERNAL format representations for RDMS are applied to datetime data. For more information, see the *Relational Database Server for ClearPath OS 2200 SQL Programming Reference Manual*.

Datetime data must meet the following criteria:

- DATE data must use the 10-character form YYYY-MM-DD.

  The following DATE value is valid:

  ```
  1998-08-11
  ```

- TIME data must minimally use the 8-character form HH:MM:SS. If the column definition includes precision within the second, specify its digits after the decimal point as follows:

  ```
  21:14:35.34356
  ```

- TIMESTAMP data must use the 19-character form YYYY-MM-DD HH:MM:SS as a minimum. Like TIME data, specify the decimal point and digits after the decimal point as follows:

  ```
  1998-08-11 21:14:35.3435
  ```

In TIME and TIMESTAMP data, if the length of digits following the decimal point is not equal to the corresponding column definition, XRLOAD processes it as follows:

Digits	XRLOAD
> definition	Ignores the surplus digits
< definition	Assumes "O" for the lack of digits

3.3.5. Rules for BLOB Data

BLOB data must meet the following criteria:

- BLOB data must use the 24-column field. That is, when the FORMAT clause is specified, the start-column and end-column must describe a 24-column field. When the FORMAT clause is omitted, XRLOAD assumes a 24-column field for a BLOB column.
- This 24-column field for BLOB data is used to store the LOB identifier and not to store BLOB text.
- The LOB identifier must be the one created by the RDMS UNLOAD statement with WITHOUT BLOB VALUES clause. You must not change the value of this LOB identifier.
- If the NULLIF clause is specified for a BLOB column to load a null value, the pattern field cannot start with a number.

Examples

The following examples illustrate a table definition, the input file from the table definition in EXTERNAL format, and the loading of the input file into the table:

The table definition in RDMS extended syntax is as follows:

```
CREATE TABLE tbl IN area
COLUMNS   c1 DECIMAL(5,1),
          c2 CHARACTER(5),
          c3 DECIMAL(5,2)
PRIMARY KEY pk IS c1 ASC
INDEX ix1 ON c2 DESC;
```

The input file in EXTERNAL format is as follows:

```
-004.0 AAAAA -04.00
-003.0 AAAAA  -3.00
-002.0 AAA    -2.00
-001.0 AAAA  -01.00
+000.0 AAAAA 000.00
+001.0 AAAA    0.00
+002.0 AAA     1.00
+003.0 ZZZZZ +02.00
+ZZZ.Z CCCCC ******
```

To load the input file into the table, enter

```
load external file " load-xr "
  sorted yes core 260k load factor 67
  into table tbl
  format    c1 (1:6)   nullif  (c3='******'),
            c2 (8:12)  nullif  ((8:9) =' z '),
            c3 (14:19) nullif  (c3='******');
```

3.4. INTERNAL Format Files

XRLOAD accepts INTERNAL format input files unloaded by the RDMS UNLOAD statement. Performance can be improved by loading data from INTERNAL format input files. INTERNAL format files bypass data conversion and semantic validation.

SDF files can also be created by following the guidelines in this subsection.

You are responsible for creating an INTERNAL format input file and ensuring that the file meets the RDMS INTERNAL format representation.

3.4.1. Input File Requirements

INTERNAL format input files must meet the following requirements:

- Each value must use the representation used by RDMS in the database.
- In the following cases, each column value must appear in accordance with the sequence of the CREATE TABLE statement.
  - INDEX clause is not specified in the LOAD command
  - FORMAT clause is not specified in the LOAD command using INDEX clause
- Records cannot contain dropped columns.
- +0 must be used to represent zero. Do not use -0.
- XRLOAD does not check the null indicator if the column is defined as NULL NOT ALLOWED. If N is set to the null indicator for the column defined NULL NOT ALLOWED, XRLOAD does not load a null value.

3.4.2. UCS COBOL Data Types

The following table compares the RDMS column definition of the valid data types for INTERNAL format input files with corresponding UCS COBOL data types:

Column Definition Length (*n*) or Precision and Scale (*p,s*)	**Corresponding UCS COBOL Data Type**
CHARACTER(*n*)	PIC X(*n*)
NCHARACTER(*n*)	PIC X(*n*) DISP-2
DECIMAL(*n,s*)[unowned schema]	PIC S9(*n-s-1*)V9(*s*) SIGN LEADING SEPARATE
DECIMAL(*p,s*)[owned schema]	PIC S9(*p-s*)V9(*s*) SIGN LEADING SEPARATE
NUMERIC(*p,s*)(*p*≤11)†*pp* See the *Relational Database Server for ClearPath OS 2200 SQL Programming Reference Manual.*	PIC S9(10-*s*)V9(*s*) BINARY
NUMERIC(*p,s*)(*p*>11)*	PIC S1(72) BINARY-1 **
INTEGER*	PIC S9(10) BINARY
SMALLINT*	PIC S9(5) BINARY
REAL	COMP-1
DOUBLE PRECISION	COMP-2
FLOAT(*p*≤27)	COMP-1
FLOAT(*p*>27)	COMP-2
DATE, TIME, TIMESTAMP	See the tables in the following subsections.
BLOB(n)	PIC X(24)***

* Not every value can be represented in other programming languages for NUMERIC, INTEGER, and SMALLINT. Moreover, a different storage assignment exists between RDMS and other programming languages. In the RDMS internal representation, SMALLINT always uses 18 bits (2 characters), NUMERIC($p \leq 11$) and INTEGER always use 36 bits (4 characters), and NUMERIC($p > 11$) always assigns 72 bits (8 characters). For the available value range for each data type, see the *Relational Database Server for ClearPath OS 2200 SQL Programming Reference Manual*.

** For more details, see A.2.

*** This field is used to store LOB identifier and not to store BLOB text.

Note: *For comparable RDMS and UCS COBOL data definitions, see Appendix A.*

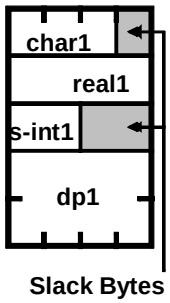
3.4.3. REAL, DOUBLE PRECISION, and FLOAT Data Types

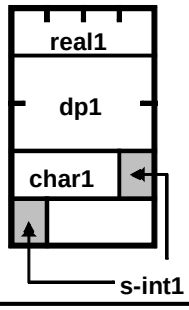
For REAL, DOUBLE PRECISION, and FLOAT column definitions in ASCII COBOL, define columns without slack bytes for fewer problems.

COMP-1 and COMP-2 data types are automatically word aligned in ASCII COBOL.

The following illustrates an incorrect and a correct definition:

	Column Definition	File Section in ASCII COBOL	Structure of write-rec

Incorrect Definition	`CREATE TABLE ... COLUMNS ARE`
	`  col1  :  CHAR(3),`
	`  col2  :  REAL,`
	`  col3  :  SMALLINT,`
	`  col4  :  DOUBLE PRECISION`
	`  .`
	`  .`
	`  .`
		`01 write-rec.`
		`  03 char1  PIC X(3).`
		`  03 real1  COMP-1.`
		`  03 s-int1 PIC 9(5) COMP.`
		`  03 dp1    COMP-2.`
			

Correct Definition	`CREATE TABLE ... COLUMNS ARE`
	`  col1  :  REAL,`
	`  col2  :  DOUBLE PRECISION,`
	`  col3  :  CHAR(3),`
	`  col4  :  SMALLINT`
	`  .`
	`  .`
	`  .`
		`01 write-rec.`
		`  03 real1  COMP-1.`
		`  03 dp1    COMP-2.`
		`  03 char1  PIC X(3).`
		`  03 s-int1 PIC 9(5) COMP.`
			

3.4.4. DATE Data Type Assignment

COBOL assigns 36 bits to the DATE data type, as follows:

Part	Bits
Year	18
Month	9
Date	9

3.4.5. TIME Data Type Assignment

COBOL assigns 36 bits to the TIME data type if the precision is less than 4 (with each field declared as an unsigned integer), as follows:

Part	Bits
Hour	9
Minutes	9
Seconds	18

COBOL assigns 72 bits to the TIME data type if the precision is 4 to 6, as follows:

Part	Bits
Hour	9
Minutes	9
Unused (cannot contain a value; must be filled with 0_2)	18
Seconds	36

3.4.6. TIMESTAMP Data Type Assignment

COBOL assigns 72 bits to the TIMESTAMP data type if the precision is less than 4 (with each field declared as an unsigned integer), as follows:

Part	Bits
Year	18
Month	9
Date	9
Hour	9
Minutes	9
Seconds	18

COBOL assigns 108 bits to the TIMESTAMP data type if the precision is 4 to 6, as follows:

Part	Bits
Year	18
Month	9
Date	9
Hour	9
Minutes	9
Unused (cannot contain a value; must be filled with 0_2)	18
Seconds	36

3.4.7. BLOB Data Type

BLOB data must meet the following criteria:

- BLOB data must use the 24-column field.
- The 24-column BLOB data field is used to store the LOB identifier and not to store BLOB text.
- The LOB identifier must be the one created by the RDMS UNLOAD statement with WITHOUT BLOB VALUES clause. You must not change the value of this LOB identifier.

3.4.8. Null Indicators

A list of null indicators must be supplied for each record in the input file if any column is defined as NULLS ALLOWED. You do not have to supply null indicators for final columns in a table defined as NULL NOT ALLOWED. In other words, a list of null indicators is not necessary when every column in the table to be loaded is defined as NULL NOT ALLOWED. If the FORMAT clause is specified in the LOAD command using the INDEX clause, null indicators must be supplied in accordance with the sequence of columns in the FORMAT clause.

Null indicators use one character for each column and must begin immediately following the last value of the record:

- An N means that the corresponding value is null.
- A tilde (~) means that the corresponding value is not null.

Example

This example illustrates a table definition and a record structure, with column C2 of the record as null data.

The table definition is as follows:

```
CREATE TABLE SCH.TBL IN SCH.ST
  COLUMNS ARE
    C1 : DECIMAL(5,1),
    C2 : CHARACTER(5),
    C3 : CHARACTER(5) NOT NULL
  PRIMARY KEY PK IS C1 ASC
  INDEX IX1 ON C2 DESC;
```

The record structure is as follows:

a	b	c	d	e
+0007	ZZZZZ	AAAAA	~	N

where:

a is the data of column C1.

b is the data of column C2.

c is the data of column C3.

d is the null indicator of column C1.

e is the null indicator of column C2.

Examples

This example illustrates a table definition, the input file from the table definition in INTERNAL format, and the loading of the input file into the table.

The table definition in an unowned schema in RDMS extended syntax is as follows:

```
CREATE TABLE tbl IN area
COLUMNS   c1 DECIMAL(5,1),
          c2 CHARACTER(5),
          c3 DECIMAL(5,2)
PRIMARY KEY pk IS c1 ASC
INDEX ix1 ON c3 DESC;
```

The input file in INTERNAL format is as follows:

```
-0040AAAAA-0400~~~
-0030AAAAA-0300~~~
-0020AAA  -0200~~~
-0010AAAA -0100~~~
+0000AAAAA+0000~~~
+0010AAAA +0000~~~
+0020AAA  +0100~~~
+0030     +0000~NN
+0000CCCCC+0000N~N
```

To load the input file into the table, enter

```
load internal file "inputfile"
  sorted yes core 260k load factor 67
  into table tbl;
```

3.5. USER Format Files

XRLOAD, like RDMUTL, accepts USER format input files. USER and INTERNAL format files have the same format except for how null indicators are specified.

3.5.1. Null Indicators

A null indicator for all columns does not need to be supplied. Use an N as a null indicator only in those cases where a null value is to be loaded in a column. If a character other than an N is specified or a null indicator is not supplied, XRLOAD loads a nonnull value in the corresponding column.

Null indicators use one character for each column in a table and must begin immediately following the last value of the record.

- An N means that the corresponding value is null.
- A character other than N or the lack of a null indicator means that the corresponding value is not null.

Example

The following example illustrates a table definition and three record structures. In all of these structures, nonnull data is loaded into columns C1 and C3, and null data is loaded only into column C2.

The table definition is as follows:

```
CREATE TABLE SCH.TBL IN SCH.ST
  COLUMNS ARE
    C1 : DECIMAL(5,1),
    C2 : CHARACTER(5),
    C3 : CHARACTER(5),
    C4 : DECIMAL(5,1) NOT NULL
  PRIMARY KEY PK IS C1 ASC
  INDEX 1X1 ON C2 DESC;
```

The record structure is as follows:

	a	b	c	d	e	f	g
structure-1	+0007	ZZZZZ	AAAAA	+0103	A	N	~
structure-2	+0008	LLLLL	YYYYY	+0102		N	A
structure-3	+0009	FFFFF	QQQQQ	+0101	~	N	

where:

a is the data of column C1.

b is the data of column C2.

c is the data of column C3.

d is the data of column C4.

e is the null indicator of column C1.

f is the null indicator of column C2.

g is the null indicator of column C3, which is not supplied in structure-3.

Section 4
Loading the Data

This section explains how to actually load the data. It explains how to call and interact with XRLOAD. It also illustrates the syntax of the LOAD command and describes its parameters.

4.1. Calling XRLOAD

Call XRLOAD before entering the LOAD command to load data by entering Executive Control Language (ECL) statements.

4.1.1. Runstream Model

Use the following series of ECL statements to call XRLOAD:

```
1.  @USE                    use-name-1.,file-name-1.
2.  [@use                    use-name-2.,file-name-2.]
        .
        .
        .
3.  @USE                    XR$SORT.,SYS$LIB$*SORT.
4.  @USE                    ABS.,file-name-3
5.  @asg,t                  sort-scratch-file-1.
6.  @asg,t                  sort-scratch-file-2.
7.  @asg,t                  sort-scratch-file-3.
8.     .
9.     .
10.    .
11.  @asg,t                  xr$select.,f///262143
12.  @asg,t                  xr$sortout.,f///262143
13.  @use                    use-name-3.,file-name-4.
14.  @ABS.XRLOAD[,options]   application-name[,use-name-4.[element-name]]
15.         ... Enter a single LOAD command here ...
16.  @eof
```

4.1.2. Notation Conventions

The preceding runstream model uses the following conventions:

- Required statements are in uppercase letters:

  ```
  @USE XR$SORT.,SYS$LIB$*SORT.
  ```

- Optional statements are in lowercase letters:

  @use *use-name-1.,file-name-1.*

- Names of items that must be supplied (or variables) are in lowercase *italic* letters:

  file-name-1

4.1.3. XRLOAD Calling Runstream Lines

The following table describes the lines of the XRLOAD runstream:

Line	Description
1-2	If the input file name is different from the internal input file name specified on the LOAD command or the file name exceeds 30 characters, use the @USE command to attach the internal input file name to the external input file name. If there are input files, use the @USE command for these input files. If the input file is not assigned, XRLOAD automatically assigns it exclusively.
3	You must use the @USE command to direct XRLOAD to a valid SORT processor file in cases where • The table to be loaded contains one or more secondary indexes and unique constraints in loading without the INDEX clause • The input file is not sorted in the table's primary key order when loading without the INDEX clause, or the specified index key when loading with the INDEX clause XRLOAD expects the SORT processor to reside in file XR$SORT. In the preceding model runstream, the SORT processor file resides in file SYSLIB*SORT.
4	Use the @USE command to direct XRLOAD to a valid JETLOAD and PTLOAD file. XRLOAD expects its storage manager JETLOAD and PTLOAD to reside in file ABS. In the following example, JETLOAD and PTLOAD resides in file SYSLIB*XRLOAD: `@use abs,sys$lib$*xrload.`
5–10	Allocate scratch files for the SORT processor to improve sort processing performance. If these statements are omitted, the SORT processor automatically allocates these scratch files at a predetermined size. If the table is partitioned, XRLOAD expects that the scratch files (XA, XB, and XC) are all assigned or that none of the scratch files (XA, XB, and XC) are assigned. A subset of these files cannot be assigned. For more information, see the *Sort/Merge Programming Guide*.

Line	Description
11	Explicitly allocate the work file XR$SELECT. JETLOAD and PTLOAD places into file XR$SELECT the items used for • Building secondary index and unique constraint B-trees when the INDEX clause is not specified. • Loading from an EXTERNAL format input file that is not sorted in the table's primary key order when the INDEX clause is not specified or the specified index key order when the INDEX clause is specified. If this statement is omitted, XRLOAD automatically allocates the file as the minimum size equal to the highest numbered granule written of the input file.
12	Explicitly allocate the work file XR$SORTOUT. The SORT processor places the sort result into file XR$SORTOUT. JETLOAD and PTLOAD also uses this file as input for • Building secondary index and unique constraint B-trees when the INDEX clause is not specified. • Loading from a file that is not sorted in the table's primary key order when the INDEX clause is not specified or the specified index key order when the INDEX clause is specified. If this statement is omitted, XRLOAD automatically allocates the file as the minimum size equal to the highest numbered granule written of the input file.
13	If the name of the file into which you want XRLOAD to store the generated runstream is different from the internal file name specified on the processor call statement, use the @USE command to attach the internal file name to the external file name. If the file is not assigned, XRLOAD automatically assigns it exclusively.
14	Call XRLOAD, where *options* are C - Set the run condition word based on Unisys standards. E - Echo on to the symbiont file. K - Bypass key compression of key items. L - Display the generated runstream in the symbiont file. N - Do not display messages at the system console for each B-tree loaded. P - Produce a postmortem dump (PMD) when an error has occurred. Z - Display the trace. application-name - The application group name to which the table to be loaded belongs. An alias can be used for the application group name. The alias cannot be longer than 6 characters. use-name-4 - The internal file name in which XRLOAD stores the generated runstream for a LOAD command. If omitted, TPF$ is assumed. element-name - The element name with which XRLOAD generates a runstream for the LOAD command. If omitted, $XRLOAD$ is assumed. A version name cannot be specified.
15	Enter a LOAD command. For more information, see 4.2.
16	Enter the end-of-file marker.

Example

The following example uses the statements on lines 3, 4, and 14 (the only three required statements) to call XRLOAD:

```
@use inputfile.,xrload*input-f.
@use xr$sort.,sys$lib$*sort.
@use abs.,sys$lib$*xrload.
@use savefile.,xrload*save-f.
@abs.xrload,e udssrc,savefile.
  load external file "inputfile"
    .
    .
    .
@eof
```

4.2. Using the LOAD Command

Use the LOAD command to load data into an RDMS database after calling XRLOAD (see 4.1).

4.2.1. Syntax

Use the following syntax:

```
LOAD [ EXTERNAL | INTERNAL | USER ]
   FILE input-file-name1[[,input-file-name2]...]
    [ SORTED { YES | NO } ] [ CORE core-parameter-for-sort ]
    [ LOAD FACTOR integer ]
    INTO [ TABLE ] table-specification
    [ INDEX index-name [ PARTITION partition-name[, partition-name] . . . ]]
    [ FORMAT item-position [,item-position ] . . . ] ;
```

4.2.2. Notation Conventions

The syntax uses the following conventions:

- Keywords are in uppercase letters with default values in **boldface** type (**EXTERNAL**).
- Names of items that must be supplied (or variables) are in lowercase *italic* letters (*input-file-name*).
- Optional items are enclosed in brackets ([**EXTERNAL** | INTERNAL | USER]). Select at least one item, each of which is separated by a vertical bar.
- Required syntax is enclosed in braces ({ YES | NO }). Choose at least one of the two or more choices, each of which is separated by a vertical bar.
- Items that can be repeated are followed by ellipses (*item-position* . . .).

4.2.3. LOAD Command Parts

The following table describes the parts of the LOAD command:

Part	Description
LOAD	The command to load data from a file.
EXTERNAL	(default) Means that the input file is in XRLOAD EXTERNAL format (that is, pure ASCII text format).
INTERNAL	Means that the input file is in XRLOAD INTERNAL format.
USER	Means that the input file is in XRLOAD USER format.
FILE *input-file-name1[,input-file-name2], . . .]*	The name of the input file to be loaded. If it contains a file qualifier, a file cycle, a read key, a write key, or a period, or if it starts with a hyphen, a dollar sign, or a number, enclose it in quotation marks. The input file name cannot exceed 30 characters (not including the period). If your input file name is longer or different from the internal input file name specified on the LOAD command, use the @USE command to attach the internal input file name to the external input file name. Here are some examples of correctly formatted file names: my-file "myqual*myfile." "myqual*myfile(-2)/rkey/wkey" Note whether the file has read or write keys. If the keys are incorrect, XRLOAD aborts. If the input file is not assigned when XRLOAD is called, XRLOAD assigns the file before loading the data. If there are input files on the FILE clause, XRLOAD loads these files into a table.
SORTED	YES - The lines in the file are already sorted in the primary key order of the tables being loaded. This enables XRLOAD to skip the time-consuming step of sorting the rows as they are being loaded. If YES is specified and XRLOAD encounters a row that is out of order, XRLOAD returns an error. NO - (default) The lines in the file are not sorted in the primary key order of the tables being loaded. Using the SORT processor, XRLOAD sorts the rows into primary key order as they are being loaded.

Part	Description
CORE *core-parameter-for-sort*	The CORE parameter in the SORT processor call (which is included in a generated runstream). Applicable only if the table to be loaded contains one or more secondary indexes and unique constraints. The CORE parameter determines the word size of the work area for the SORT processor (an integer no larger than 99,999,999). Use K for 10^3 or M for 10^6. If omitted, the SORT processor uses its default core size.
LOAD FACTOR *integer*	The loading factor for B-tree pages, both data and index pages (including secondary index pages). Its value can be between 50 and 100. If omitted, 100 is assumed.
INTO [TABLE] *table-specification*	The name of the table to be loaded in the format [*qualifier.*]*table-name*[*:version-name*] If the qualifier is omitted, RDMS is assumed. If the version name is omitted, PRODUCTION is assumed.
INDEX *index-name*	If used, loads B-tree independently. If omitted, load into the table is assumed.
PARTITION *partition-name*	If used, loads specified partitions. If omitted, load into all partitions that constitute the specified index. ***Note:*** *If the PARTITION clause is specified, the records that are not within the specified partitions are ignored. In other words, you can use the same input file that is for loading to the table.*
FORMAT *item-position, ...*	If omitted, XRLOAD assumes that • The input file contains all columns of the table. • The columns value appears in the sequence implied from the table definition. • Every data value is in the representation implied from the column data type and the file format (INTERNAL, EXTERNAL, or USER format). • Each column data is delimited by a single ASCII blank. If used, there are two formats for item-position. One is for an EXTERNAL format file, another is for an INTERNAL or a USER format file. Note that this clause can be used for an INTERNAL or a USER format file only when the INDEX clause is used. See 4.2.5 for the format of the *item-position* field.

4.2.4. FILE Part Considerations for Multiple Inputs

When multiple inputs are specified, the following considerations apply:

- The maximum number of input files is 10. XRLOAD issues errors when the number of input files exceeds 10.
- The file TYPE must be the same. For example, if the EXTERNAL Format file is used, all inputs must be EXTERNAL Format files.
- If the input file is made by the UNLOAD command with DESCRIPTION records, XRLOAD allows only the first file.
- Some TAPE files are not specified on the FILE clause. Only one TAPE file is allowed.
- Input files are merged into XR$SELECT, XR$SORTOUT files. These files do not exceed the maximum file size on the OS 2200 system.

Example syntax for multiple inputs follows:

```
load external file "input1", "input2"  sorted into sch.tbl1 ;
```

4.2.5. Format of Item-Position Field

The format of the *item-position* field for an EXTERNAL format file is

column-name [POSITION] *(start-column[:end-column])[nullif-clause]*

In this format, specify the starting position of each column in the input file.

The format for an INTERNAL or a USER format file is

column-name

In this format, specify only the order of columns in the input file. XRLOAD determines the item position and size for each column based on the column definition.

In both formats specify the columns in any order; however

- Every column in a table must be accounted for when the INDEX clause is not used.
- Every column of which the specified index B-tree is composed must be accounted for when the INDEX clause is used. In other words, if the columns in a table that do not construct the B-tree are specified, they will be ignored.

4.2.6. Item-Position Parts

The following table describes the parts of the *item-position* field:

Part	Description
column-name	The name of a column.
POSITION	An optional keyword.
start-column	The starting position of the column.
end-column	The ending position of the column. If omitted, XRLOAD automatically calculates this position from the table definition.
nullif-clause	Indicates when to load a null value from the input file into a column if the input data matches the pattern of the NULLIF clause. See 4.2.7 for the format of the NULLIF clause.

4.2.7. Format of NULLIF Clause

The format of the NULLIF clause is

```
NULLIF ( [ column-name | (null-start-column [:null-end-column]) ] ='pattern' )
```

4.2.8. NULLIF Clause Parts

The following table describes the parts of the NULLIF clause:

Part	Description
NULLIF	The NULLIF keyword.
column-name	The name of a column in the table being loaded, which must be specified in the FORMAT clause.
null-start-column	The starting position of the null indicator pattern.
null-end-column	The ending position of the null indicator pattern. If omitted, XRLOAD calculates a default value for this position from the length of the character string.
pattern	The null value pattern up to 30 characters. The length cannot exceed the assumed length calculated from one of the following: • The *null-start-column* and *null-end-column* • The assumed length calculated from the *start-column* and *end-column* item position If the length is too short, XRLOAD fills the remaining character positions with ASCII blank characters to match the input file records. The null check uses character patterns. Therefore, patterns 4 and +004 are treated differently (even though they are numerically equivalent) because XRLOAD compares patterns and not values. If the NULLIF clause is specified for a BLOB column, the pattern field cannot start with a number.

Example

The following example loads EXTERNAL format file LOAD-XR with a load factor of 80 into table SCMRN.TBL01. It is loading five columns with the starting and ending positions for each column indicated in parentheses. The last three columns are to be loaded with null values if they match the indicated pattern.

```
load external file "load-xr"
sorted yes
load factor 80
into scmrn.tbl01
  format
  pkey   ( 1:10),
  pkey_2 (12:20),
  skey_1 (22:26) nullif (skey_1='zzzzz'),
  skey_2 (28:31) nullif ((1:1) = '9'),
  nkey_1 (33:37) nullif (skey_1 = 'zzzzz');
```

Section 5
The Process

This section explains the XRLOAD process from the moment the LOAD command is entered to the completion of the loading process and the return of output messages. It also offers tips for improving performance.

5.1. How XRLOAD Loads Data

The XRLOAD processor and its storage manager JETLOAD work together to load the data and return information and messages.

Examples

See Appendix B for sample runstreams.

5.1.1. The XRLOAD Process

The following table describes the 5-step XRLOAD loading process into a nonpartitioned table after a LOAD command is entered:

Step	The XRLOAD Processor	And if...	Then XRLOAD...
1	Parses the command and checks the syntax.	A syntax error is detected	Returns an error message and terminates the process.
2	Generates a runstream and saves it using the file and element name specified on the XRLOAD processor call statement.	A file and element name were not specified	Uses the default file and element name.
3	Generates an element that contains information about the table and storage area definition. XRLOAD garners this information from the application group's FDT$ and RDT$FILE files.		

Step	The XRLOAD Processor	And if...	Then XRLOAD...
4	Automatically adds the generated runstream to start the loading process.	The L option was specified on the processor call statement	Returns the generated runstream in the current symbiont file before adding it.
5	Repeatedly calls JETLOAD and the SORT processor for each B-tree it is loading. When the input file is empty, JETLOAD loads all B-trees at once. This means JETLOAD initializes all B-tree storage areas of the table to be loaded.	An EXTERNAL format input file is not sorted in the table's primary key order	Calls JETLOAD one more time before loading the primary key B-tree.

For loading into partitioned tables, see Appendix D.

5.1.2. JETLOAD Output

JETLOAD returns the following information in the current symbiont file about each B-tree as it is loaded:

- Number of records loaded into each B-tree
- Name of the schema
- Name of the B-tree storage area
- Number of data pages written in the B-tree
- Number of index pages written in the B-tree
- Highest page number allocated for the B-tree in the storage area
- Number of B-tree levels

Note: *When the input file is empty, JETLOAD loads all B-trees at once and returns the information on all storage areas in the current symbiont file. For further information, see Appendix C.*

5.1.3. JETLOAD Completion Messages

JETLOAD returns a completion message for each B-tree loaded

- To the current symbiont file
- To the console, unless blocked by the N option on the XRLOAD processor call statement

5.2. Improving Load Performance

The performance of XRLOAD can be improved by making changes to the table description, the storage area, the input file, and the execution of XRLOAD.

For special considerations when loading into partitioned tables, see Appendix D.

5.2.1. Table Description

To make changes to the table description if the input file is in INTERNAL or USER format:

- Define primary key columns first.
- Define primary key columns continuously in primary key order.
- Define the columns with numeric data types continuously.
- Wherever null values are not to be stored in a column, use the NOT NULL column constraint.

5.2.2. Input File

To make changes to the input file:

- Use INTERNAL format input files to reduce CPU usage.
- Use shorter patterns in the NULLIF clause to reduce instructions for comparisons on null checks.

  This technique is available only for nonindexed items because of additional requirements regarding NULLIF clauses for primary key and secondary index columns as described in 3.3.
- Allocate the decimal point at a fixed location in each record in the input file if the input file is in EXTERNAL format and numeric data including the decimal point are loaded.
- Sort the records in the table's primary key order and use the SORTED YES clause on the LOAD command.
- For INTERNAL and USER format, do not supply null indicators for final columns in a table defined as NOT NULL.
- For USER format, if null values are not to be loaded into final columns in a table defined as NULL ALLOWED in all input records, do not supply null indicators for those columns.
- Supply a formal null indictor (a tilde [~] or N) for those columns into which a null value may be loaded, even for USER format.

5.2.3. Execution of XRLOAD

To make changes to the execution:

- Optimize the size of sort scratch files and the SORT processor's CORE parameter to achieve better SORT processor performance.

  XRLOAD uses the SORT processor to presort records for

  – Loading into secondary index and unique constraint B-trees.

  – Loading from a file that is not sorted in the table's primary key order.

  For further information, see the *Sort/Merge Programming Guide*.

- Optimize the size of the XR$SELECT and XR$SORTOUT intermediate files for XRLOAD/JETLOAD.

  For XR$SELECT, in either of the following two cases:

  – Use the SORTED YES clause on the LOAD command.

  – Use the SORTED NO clause on the LOAD command and an INTERNAL or USER format input file.

    Determine the initial reserve size in tracks (TRK) using the following formula:

    ```
    TRK = ( ( (A + B +3)//4 ) * C + 1791 )//1792
    ```

    where:

    A is the sum of characters in primary keys, secondary indexes, and unique constraints.

    B is the number of columns in primary keys, secondary indexes, and unique constraints.

    C is the number of records.

  For XR$SORTOUT, if the SORTED YES clause on the LOAD command is used:

  1. Calculate the size of secondary indexes and unique constraints.
  2. Select the secondary index or unique constraint with the largest size.

3. Determine the initial reserve size in tracks (TRK) using the following formula:

```
TRK = ( ( (A1 + A2 + B1 + B2 + 3)//4 ) * C + 1791 )//1792
```

where:

A1 is the sum of characters in the primary key.

A2 is the sum of characters in the secondary index or unique constraint.

B1 is the number of columns in the primary key.

B2 is the number of columns in the secondary index or unique constraint.

C is the number of records.

In the following two cases:

– For XR$SELECT if the SORTED NO clause on the LOAD command and an EXTERNAL format input file is used.
– For XR$SORTOUT if the SORTED NO clause on the LOAD command is used.

Determine the initial reserve size in tracks (TRK) using the following formula:

```
TRK = ( ( (A + B +3)//4 ) * C + 1791 )//1792
```

where:

A is the sum of all the characters in the table.

B is the number of all the columns in the table.

C is the number of records.

- Allocate the disks used by XRLOAD on faster disk units to improve overall performance.
- Specify the exact data size of the item-position field on the LOAD command.

  Do this if the input file is in EXTERNAL format and the exact data size of the following columns is shorter than the table definition size:

  – Character columns are included in the primary key, secondary index, or unique constraint.
  – Any character columns in the input file that are not sorted on the table's primary key order.
- Use the SORTED YES clause on the LOAD command if the input file is sorted in the table's primary key order.
- Use the W option on the @RUN image to avoid unnecessary I/Os in normal cases.

Appendix A
UCS COBOL Data Definitions

This appendix illustrates comparable RDMS and UCS COBOL data definitions.

Notation Conventions

For DECIMAL data in a column definition for a table in an unowned schema, the two values in parentheses following the data type declaration are precision+1 and scale (n,s), as in the following example, where 15 characters includes the leading sign:

```
a:DECIMAL(15,3)
```

For DECIMAL data in a column definition for a table in an owned schema, the two values in parentheses following the data type declaration are precision and scale (p,s), as in the following example, where 15 characters exclude the leading sign:

```
a:DECIMAL(15,3)
```

For NUMERIC data, the two values in parentheses following the data type declaration are precision and scale (p,s), as in the following example:

```
a:NUMERIC(8,3)
```

For TIME and TIMESTAMP data, the single value in parentheses following the data type declaration is precision (p), as in the following example:

```
a:TIME(6)
```

A.1. CHARACTER and NCHARACTER Data

Specify the same character length in WORKING-STORAGE SECTION as is specified in the column definition.

The following table illustrates comparable RDMS CHARACTER, NCHARACTER, and UCS COBOL data types:

Column Definition	Working Storage Section
a:CHARACTER(5), b:NCHARACTER(4)	01 rec. 03 char PIC X(5). 03 nchr PIC X(4) DISP-2.

A.2. DECIMAL, INTEGER, SMALLINT, and NUMERIC Data

A.2.1. DECIMAL Data

Use the SIGN LEADING SEPARATE form and USAGE DISPLAY. If USAGE DISPLAY is omitted, it is assumed by default; it is included in this example for clarity.

Unowned Schema

The following table illustrates comparable RDMS DECIMAL and UCS COBOL data types for a table in an unowned schema:

Column Definition	Working–Storage Section
a:DECIMAL(15,3)	01 rec. 03 dec PIC S9(11)V9(3) SIGN LEADING SEPARATE USAGE DISPLAY.

Owned Schema

The following table illustrates comparable RDMS DECIMAL and UCS COBOL data types for a table in an owned schema:

Column Definition	Working–Storage Section
a:DECIMAL(15,3)	01 rec. 03 dec PIC S9(12)V9(3) SIGN LEADING SEPARATE USAGE DISPLAY.

A.2.2. INTEGER Data

Use a 4-character unscaled binary.

The following table illustrates comparable RDMS INTEGER and UCS COBOL data types:

Column Definition	Working–Storage Section
a:INTEGER	01 rec. 03 int PIC S9(10) BINARY.

A.2.3. SMALLINT Data

Use a 2-character unscaled binary.

The following table illustrates comparable RDMS SMALLINT and UCS COBOL data types:

Column Definition	Working–Storage Section
a:SMALLINT	01 rec. 03 int PIC S9(5) BINARY.

A.2.4. NUMERIC Data

NUMERIC data depends on whether the precision is less than 12 or greater than 11.

Precision Less than 12

Use a 4-character binary with as much scale as needed.

The following table illustrates comparable RDMS NUMERIC of precision less than 12 and UCS COBOL data types:

Column Definition	Working–Storage Section
a:NUMERIC(8,3)	01 rec. 03 num PIC S9(7)V9(3) BINARY.

Precision Greater than 11

UCS COBOL does not allow 8-character scaled binary. Use an unscaled binary instead to assign 8 characters, as follows:

```
S1(72) BINARY-1.
```

And then use an additional variable to represent the value you want to load, as follows:

```
S9(18-s)V9(s) BINARY.
```

where *s* stands for scale.

The following table illustrates comparable RDMS NUMERIC of precision greater than 11 and UCS COBOL data types:

Column Definition	Data Division
For writing to the file. a:NUMERIC(16,3)	01 write-rec. . . . 03 num1 PIC S1(72) BINARY-1. . . .
For the intermediate variable.	01 cal. 03 cal2 PIC S9(15)V9(3) BINARY. 03 mv1 REDEFINES cal2 PIC S9(18) BINARY. PROCEDURE DIVISION. MOVE -123.456 TO cal2. MOVE mv1 TO num1. WRITE write-rec. STOP RUN.

A.3. REAL, DOUBLE PRECISION, and FLOAT Data

For REAL and FLOAT of precision less than 28, use COMP-1.

For DOUBLE PRECISION and FLOAT of precision greater than 27, use COMP-2.

See also 3.4.

A.4. DATE, TIME, and TIMESTAMP Data

A.4.1. Comparable UCS and RDMS Datetime Data Types

Universal Compiling System (UCS) languages have no equivalent to RDMS datetime data types.

A.4.2. DATE Data

The following table illustrates comparable RDMS DATE and UCS COBOL data types:

Column Definition	Working–Storage Section
a:DATE	01 date1. 03 year PIC 9(5) BINARY. 03 mon PIC 9(2) BINARY. 03 ddd PIC 9(2) BINARY.

A.4.3. TIME Data

TIME data depends on whether the precision is less than 4 or greater than 3.

Precision Less than 4

The following table illustrates comparable RDMS TIME of precision less than 4 and UCS COBOL data types:

Column Definition	Working–Storage Section
a:TIME(2)	01 time1. 03 hour PIC 9(2) BINARY. 03 minu PIC 9(2) BINARY. 03 secn PIC 9(3)V9(2) BINARY. (*that is*, 03 secn PIC 9(5-p)V9(p) BINARY.)

Precision Greater than 3

The following table illustrates comparable RDMS TIME of precision greater than 3 and UCS COBOL data types:

Column Definition	Working–Storage Section
a:TIME(6)	01 time2. 03 hour PIC 9(2) BINARY. 03 minu PIC 9(2) BINARY. 03 fl PIC 9(5) BINARY VALUE ZERO. 03 secn PIC 9(4)V9(6) BINARY. (*that is*, 03 secn PIC 9(10-p)V9(p) BINARY.)

A.4.4. TIMESTAMP Data

TIMESTAMP data depends on whether the precision is less than 4 or greater than 3.

Precision Less than 4

The following table illustrates comparable RDMS TIMESTAMP of precision less than 4 and UCS COBOL data types:

Column Definition	Working–Storage Section
A:TIMESTAMP(3)	01 timestamp1. 03 year PIC 9(5) BINARY. 03 mon PIC 9(2) BINARY. 03 ddd PIC 9(2) BINARY. 03 hour PIC 9(2) BINARY. 03 minu PIC 9(2) BINARY. 03 secn PIC 9(2)V9(3) BINARY. (*that is,* 03 secn PIC 9(5-p)V9(p) BINARY.)

Precision Greater than 3

The following table illustrates comparable RDMS TIMESTAMP of precision greater than 3 and UCS COBOL data types:

Column Definition	Working–Storage Section
A:TIMESTAMP(5)	01 timestamp2. 03 year PIC 9(5) BINARY. 03 mon PIC 9(2) BINARY. 03 ddd PIC 9(2) BINARY. 03 hour PIC 9(2) BINARY. 03 minu PIC 9(2) BINARY. 03 fl PIC 9(5) BINARY VALUE ZERO. 03 secn PIC 9(5)V9(5) BINARY. (that is, 03 secn PIC 9(10-p)V9(p) BINARY.)

A.5. BLOB Data

Use a 24-column field to store LOB identifier and not to store BLOB text.

The following table illustrates comparable RDMS BLOB and UCS COBOL data types:

Column Definition	Working-Storage Section
a:BLOB(50K)	01 rec. 03 blob-id PIC X(24).

Appendix B
Sample Runstreams

This appendix contains sample XRLOAD/JETLOAD runstreams.

B.1. Table Definition

The table definition is as follows:

```
CREATE TABLE SCH.TBL
  IN  SCH.STOPKD COLUMNS ARE
    COL01 : INTEGER        NOT NULL,
    COL02 : DECIMAL(16)    NOT NULL,
    COL03 : DECIMAL(16)    NOT NULL,
    COL04 : DECIMAL(16)    NOT NULL,
    COL05 : DECIMAL(16)    NOT NULL,
    COL06 : DECIMAL(16)    NOT NULL,
    COL07 : DECIMAL(16)    NOT NULL,
    COL08 : DECIMAL(16)    NOT NULL,
    COL09 : DECIMAL(16)    NOT NULL,
    COL10 : DECIMAL(16)    NOT NULL,
    COL11 : DECIMAL(16)    NOT NULL,
    COL12 : DECIMAL(16)    NOT NULL,
    COL13 : DECIMAL(16)    NOT NULL,
    COL14 : CHARACTER(20)  NOT NULL,
    COL15 : CHARACTER(20)  NOT NULL,
    COL16 : CHARACTER(20)  NOT NULL,
    COL17 : CHARACTER(20)  NOT NULL,
    COL18 : CHARACTER(20)  NOT NULL,
    COL19 : CHARACTER(20)  NOT NULL,
    COL20 : CHARACTER(20)  NOT NULL
  PRIMARY KEY PK IS COL01,COL02,COL03,COL04,COL05
    IN  SCH.STOPKI
  INDEX   IX1   ON COL09,COL10,COL11,COL12,COL13
    IN  SCH.STOIX1
  INDEX   IX2   ON COL09
    IN  SCH.STOIX2 ;
```

B.2. XRLOAD Execution Runstream

The following runstream illustrates the execution of XRLOAD:

```
@USE    DATAF.,OUT-F.
@USE    XR$SORT.,SORT$FILE.
@USE    ABS.,XRUTL.
@USE    SAVEFILE,,JCL-SAVE-F.
@XRUTL.XRLOAD,E   UDSSRC,SAVEFILE.SAVEJCL
LOAD INTERNAL FILE "DATAF" SORTED YES INTO SCH.TBL
;
@EOF
```

B.3. Runstream Generated by XRLOAD

The following runstream is generated by XRLOAD after it is executed:

```
@MSG    **** XRLOAD START ****
@ABS.JETLOAD UDSSRC,SAVEFILE,SAVEJCL,PK
@PMD,ED
@MSG    **** XRLOAD 1st Index (PK) END ( 1/ 3) ****
@XR$SORT.SORT,S
FILEIN=XR$SELECT
FILEOUT=XR$SORTOUT
SELECT=0069,0016:0085,0016:0101,0016:0117,0016:;
00000000133,0016:0001,0068
KEY=0069,0016,R,A:0085,0016,R,A:0101,0016,R,A:0117,0016,R,A:0133,0016,R,A:;
00000001,0004,B,A:0005,0016,R,A:0021,0016,R,A:0037,0016,R,A:0053,0016,R,A
@EOF
@ABS.JETLOAD UDSSRC,SAVEFILE,SAVEJCL,IX01
@PMD,ED
@MSG    **** XRLOAD 2nd Index (IX1) END ( 2/ 3) ****
@XR$SORT.SORT,S
FILEIN=XR$SELECT
FILEOUT=XR$SORTOUT
SELECT=0069,0016:0001,0068
KEY=0069,0016,R,A:0001,0004,B,A:0005,0016,R,A:0021,0016,R,A:0037,0016,R,A:;
00000053,0016,R,A
@EOF
@ABS.JETLOAD UDSSRC,SAVEFILE,SAVEJCL,IX02
@PMD,ED
@MSG    **** XRLOAD 2nd Index (IX2) END ( 3/ 3) ****
@MSG    **** XRLOAD END ****
```

B.4. Execution Result of XRLOAD

The following runstream illustrates the executed result of XRLOAD:

```
@XRUTL.XRLOAD,E   UDSSRC,SAVEFILE.SAVEJCL
XRLOAD 5R2-1A01*00 PROCESSOR  04/16/01   13:38:31
Please enter LOAD COMMAND followed by a semicolon (;)
   1.  LOAD INTERNAL FILE "DATAF" SORTED YES INTO SCH.TBL
   2.  ;
                 ***** XRLOAD  END  *****
@MSG    **** XRLOAD START ****
@ABS.JETLOAD UDSSRC,SAVEFILE,SAVEJCL,PK
JETLOAD  start  at  04/16/01 13:38:35
The total number of rows loaded is   100000
Information of storage-area  "rsm-data-only "
Schema name                   SCH
Storage-area name             STOPKD
        The total data  page is   5000  on this B-tree
        The last page number is   5001  on this B-tree
Information of storage-area  "rsm-index-only"
Schema name                   SCH
Storage-area name             STOPKI
        The total index page is     66  on this B-tree
        The last page number is     70  on this B-tree
This B-tree level is       3
JETLOAD   end   at  04/16/01 13:39:21
@PMD,ED
@MSG    **** XRLOAD 1ST INDEX (PK) END ( 1/ 3) ****
@XR$SORT.SORT,S
SORT  2OR2A SL76R204/16/01 13:39:21

FILEIN=XR$SELECT
FILEOUT=XR$SORTOUT
SELECT=0069,0016:0085,0016:0101,0016:0117,0016:;
00000000133,0016:0001,0068
KEY=0069,0016,R,A:0085,0016,R,A:0101,0016,R,A:0117,0016,R,A:0133,0016,R,A:;
00000001,0004,B,A:0005,0016,R,A:0021,0016,R,A:0037,0016,R,A:0053,0016,R,A
```

```
      100000 RECORDS IN XR$SELECT

*** SORT/MERGE SUMMARY ***

OPERATION = SORT

INPUT STATISTICS:
      100000 RECORDS IN            0 SKIPPED                     0 REJECTED
        2047 WORDS/RECORD ON INPUT
      100000 RECORDS SORTED
          74 WORDS/RECORD THROUGH SORT

OUTPUT STATISTICS:
      100000 RECORDS OUT           0 DELETED ON OUTPUT
          37 WORDS/RECORD ON OUTPUT
       14336 WORDS/BLOCK ON OUTPUT
@ABS.JETLOAD UDSSRC,SAVEFILE,SAVEJCL,IX01
JETLOAD  start  at  04/16/01 13:40:08
The total number of rows loaded is   100000
Information of storage-area  "rsm-index-only"
Schema name                   SCH
Storage-area name             STOIX1
        The total data  page is   1503  on this B-tree
        The total index page is     43  on this B-tree
        The last page number is   1548  on this B-tree
This B-tree level is       4
JETLOAD   end   at  04/16/01 13:40:20
@PMD,ED
@MSG    **** XRLOAD 2ND INDEX (IX1) END ( 2/ 3) ****
@XR$SORT.SORT,S
SORT  20R2A SL76R204/16/01 13:40:20

FILEIN=XR$SELECT
FILEOUT=XR$SORTOUT
SELECT=0069,0016:0001,0068
KEY=0069,0016,R,A:0001,0004,B,A:0005,0016,R,A:0021,0016,R,A:0037,0016,R,A:;
00000053,0016,R,A
```

```
      100000 RECORDS IN XR$SELECT

*** SORT/MERGE SUMMARY ***

OPERATION = SORT

INPUT STATISTICS:
      100000 RECORDS IN            0 SKIPPED                     0 REJECTED
        2047 WORDS/RECORD ON INPUT
      100000 RECORDS SORTED
          42 WORDS/RECORD THROUGH SORT

OUTPUT STATISTICS:
      100000 RECORDS OUT           0 DELETED ON OUTPUT
          21 WORDS/RECORD ON OUTPUT
       14336 WORDS/BLOCK ON OUTPUT
@ABS.JETLOAD UDSSRC,SAVEFILE,SAVEJCL,IX02
JETLOAD  start  at  04/16/01 13:40:50
The total number of rows loaded is   100000
Information of storage-area  "rsm-index-only"
Schema name                   SCH
Storage-area name             STOIX2
        The total data  page is   1316  on this B-tree
        The total index page is     22  on this B-tree
        The last page number is   1346  on this B-tree

This B-tree level is       3
JETLOAD   end   at  04/16/01 13:40:59
@PMD,ED
@MSG    **** XRLOAD 2ND INDEX (IX2) END ( 3/ 3) ****
@MSG    **** XRLOAD END ****
```

Appendix C
Relational Database Fast Load with Empty Input File

This appendix describes the XRLOAD process when the input file is empty.

C.1. The Process

When the input file is empty, XRLOAD initializes the storage areas. For example, assume that a company's one-day sales data is generated each day by XRLOAD. If no sales occur on a given day, the input file is empty. In this case, XRLOAD must initialize the storage areas to show that no sale occurred on that day.

Usually XRLOAD repeatedly calls JETLOAD for each individual B-tree. When the input file is empty, XRLOAD calls JETLOAD only once even though the table to be loaded contains one or more secondary indexes and unique constraints. JETLOAD initializes all storage areas at once if the table uses multifile relations.

When some input files are specified on the LOAD command, all input files must be empty to initialize the storage areas.

C.2. Table Def

The table definition is as follows:

```
CREATE TABLE SCH.TBL
    IN  SCH.STOPKD COLUMNS ARE
      COL01 : INTEGER        NOT NULL,
      COL02 : DECIMAL(16)    NOT NULL,
      COL03 : DECIMAL(16)    NOT NULL,
      COL04 : DECIMAL(16)    NOT NULL,
      COL05 : DECIMAL(16)    NOT NULL,
      COL06 : DECIMAL(16)    NOT NULL,
      COL07 : DECIMAL(16)    NOT NULL,
      COL08 : DECIMAL(16)    NOT NULL,
      COL09 : DECIMAL(16)    NOT NULL,
      COL10 : DECIMAL(16)    NOT NULL,
      COL11 : DECIMAL(16)    NOT NULL,
      COL12 : DECIMAL(16)    NOT NULL,
      COL13 : DECIMAL(16)    NOT NULL,
      COL14 : CHARACTER(20)  NOT NULL,
      COL15 : CHARACTER(20)  NOT NULL,
```

```
  COL16 : CHARACTER(20)  NOT NULL,
  COL17 : CHARACTER(20)  NOT NULL,
  COL18 : CHARACTER(20)  NOT NULL,
  COL19 : CHARACTER(20)  NOT NULL,
  COL20 : CHARACTER(20)  NOT NULL
PRIMARY KEY PK IS COL01,COL02,COL03,COL04,COL05
  IN  SCH.STOPKI
INDEX   IX1   ON COL09,COL10,COL11,COL12,COL13
  IN  SCH.STOIX1
INDEX   IX2   ON COL09
  IN  SCH.STOIX2;
```

C.3. XRLOAD Execution Runstream

The following runstream illustrates the execution of XRLOAD:

```
@USE    DATAF.,ZERO-F.
@USE    XR$SORT.,SORT$FILE.
@USE    ABS.,XRUTL.
@USE    SAVEFILE.,JCL-SAVE-F.
@XRUTL.XRLOAD,E   UDSSRC,SAVEFILE.SAVEJCL
LOAD INTERNAL FILE "DATAF" SORTED YES INTO SCH.TBL
;
@EOF
```

C.4. Runstream Generated by XRLOAD

The following runstream is generated by XRLOAD after it is executed:

```
@MSG    **** XRLOAD START ****
@ABS.JETLOAD UDSSRC,SAVEFILE,SAVEJCL,ALL
@PMD,ED
@MSG    **** XRLOAD STORAGE-AREA INITIALIZE PROCESS END ****
@MSG    **** XRLOAD END ****
```

C.5. Execution Result of XRLOAD

The following runstream illustrates the executed result of XRLOAD:

```
@XRUTL.XRLOAD,E   UDSSRC,SAVEFILE.SAVEJCL
XRLOAD 5R2-1A01*00 PROCESSOR  04/16/01   13:36:41
Please enter LOAD COMMAND followed by a semicolon (;)
   1.  LOAD INTERNAL FILE "DATAF" SORTED YES INTO SCH.TBL
   2.  ;
                  ***** XRLOAD  END  *****
@MSG    **** XRLOAD START ****
@ABS.JETLOAD UDSSRC,SAVEFILE,SAVEJCL,ALL
JETLOAD  start  at  04/16/01 13:36:45
The total number of rows loaded is        0
Information of storage-area  "rsm-data-only "
Schema name                   SCH
Storage-area name             STOPKD
        The total data  page is      0  on this B-tree
        The last page number is      1  on this B-tree
Information of storage-area  "rsm-index-only"
Schema name                   SCH
Storage-area name             STOPKI
        The total index page is      0  on this B-tree
        The last page number is      1  on this B-tree
This B-tree level is       0
The total number of rows loaded is        0
Information of storage-area  "rsm-index-only"
Schema name                   SCH
Storage-area name             STOIX1
        The total data  page is      0  on this B-tree
        The total index page is      0  on this B-tree
        The last page number is      1  on this B-tree
This B-tree level is       0
The total number of rows loaded is        0
Information of storage-area  "rsm-index-only"
Schema name                   SCH
Storage-area name             STOIX2
        The total data  page is      0  on this B-tree
        The total index page is      0  on this B-tree
        The last page number is      1  on this B-tree
This B-tree level is       0
JETLOAD   end   at  04/16/01 13:36:46
@PMD,ED
@MSG    **** XRLOAD STORAGE-AREA INITIALIZE PROCESS END ****
@MSG    **** XRLOAD END ****
```

Appendix D
Loading into Partitioned Tables

This appendix describes the XRLOAD process, and the requirements and special considerations when loading into partitioned tables.

D.1. The Process

The following table describes the 5-step general XRLOAD loading process into a table, which is partitioned after a LOAD command is entered:

Step	The XRLOAD Processor	And if...	Then XRLOAD...
1	Parses the command and checks the syntax.	A syntax error is detected	Returns an error message and terminates the process.
2	Generates a runstream and saves it using the file and element name specified on the XRLOAD processor call statement.	A file and element name were not specified	Uses the default file and element name.
3	Generates elements that contain information about the table and partition definition. XRLOAD garners this information from the application group's RDT$FILE and PART$FILE files.	An input file is empty	Returns an error message and terminates the process.

Step	The XRLOAD Processor	And if...	Then XRLOAD...
4	Automatically adds the generated runstream to start the loading process.	The L option was specified on the processor call statement	Returns the generated runstream in the current symbiont file before adding it.
5	Repeatedly calls PTLOAD and the SORT processor for each B-tree it is loading. When the INDEX clause is specified on the LOAD command, only the PTLOAD and the SORT processor for the specified B-tree are called.	An EXTERNAL format input file is not sorted in the table's primary key (loading without INDEX clause) or specified index key (loading with INDEX clause) A partition has no record to be loaded. There are some records that are not within any target partitions.	Calls PTLOAD one more time before loading the primary key B-tree (loading without INDEX clause) or specified key B-tree (loading with INDEX clause). Leaves the partition untouched. In case of loading with PARTITION clause, those records are ignored. If PARTITION clause is not specified, returns an error message and terminates the process.

D.2. PTLOAD Output

PTLOAD returns the following information in the current symbiont file about each partition as it is loaded:

- Name of the partition
- Number of records loaded into each partition
- Name of the schema
- Name of the partition storage area
- Number of data pages written in the partition
- Number of index pages written in the partition
- Highest page number allocated in the storage area for the partition
- Number of B-tree levels for the partition

PTLOAD also returns, for each key or index, the number of records loaded into the key or index.

D.3. PTLOAD Completion Messages

PTLOAD returns a completion message for each key or index loaded

- To the current symbiont file
- To the console, unless blocked by the N option on the XRLOAD processor call statement

D.4. Special Requirements for Partitioned Tables

The partition and its key definition must meet the following formula:

{ (A + 2) // 2 + (B + 3) // 4 + 3 } * C + 10 < or = 65000

where:

A is the number of columns in the partitioning key.

B is the number of characters (bytes) in the partitioning key.

C is the number of partitions.

If the above formula is not satisfied, XRLOAD returns ERROR 50029.

When the partition key consists of multiple columns, each order (ascending/descending) specified to the partition key column must be identical and if not satisfied, XRLOAD returns ERROR 50031.

D.5. Improving Load Performance

You can improve the performance of XRLOAD by making special changes to the input file for the partitioned tables.

You can make changes to the input file as follows:

- If the INDEX clause is specified
  - Presort the input records in the specified index key order.
  - Extract those columns that pertain to the specified index.
- If the PARTITION clause is specified, prepare an input file that contains only the records you want to load.

D.6. Loading into a Partitioned Table – Once

The following figure illustrates the loading process into a partitioned table by calling XRLOAD only once per table. This is the simplest loading method for a partitioned table. The same LOAD command can be used for a table that is not partitioned.

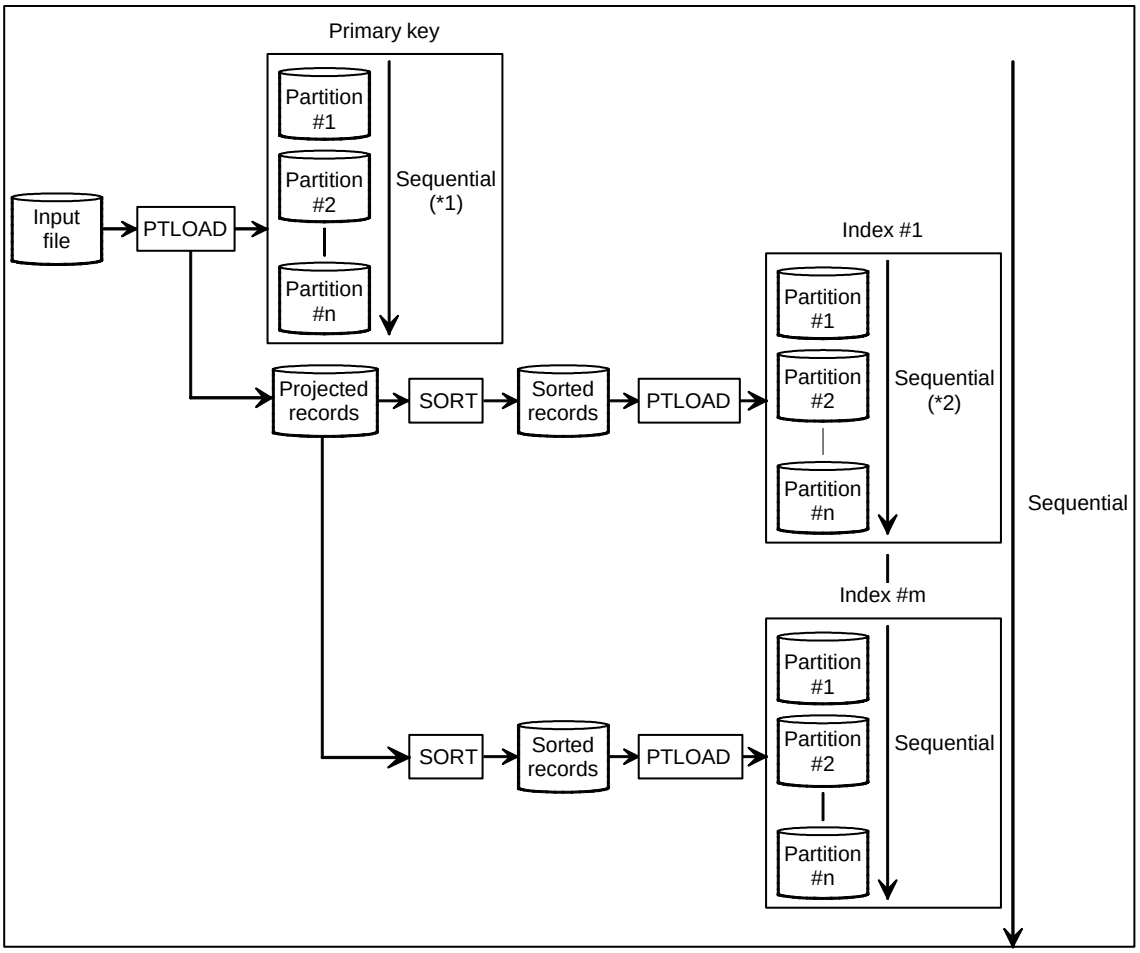


Figure D–1. Loading Process into a Partitioned Table – Once

In a generated runstream, PTLOAD first loads the input records into the corresponding partition storage area for the primary key B-tree in a sequential manner. (See *1.) And, if the table has any index (secondary indexes and/or unique constrains), PTLOAD loads the records, which are sorted in the target index key order into the corresponding index partition storage area in a sequential manner. (See *2.)

These SORT and PTLOAD processes are performed repeatedly for each secondary index and unique constraint.

D.7. Loading into a Partitioned Table – Parallel

In the interest of improving the load performance, perform a parallel load by using the INDEX clause.

The XRLOAD INDEX clause supports

- Loading from a file that is not sorted in the target index key order, if SORTED NO clause is specified. In this case, XRLOAD presorts the input rows into the target index key order before loading. Therefore, use the LOAD FACTOR clause.
- EXTERNAL, INTERNAL and USER format files, and using the FORMAT clause to any of these format files. If the FORMAT clause is omitted in the LOAD INDEX command, XRLOAD assumes that the input record is supplied by the same format as in the case that INDEX clause is not specified. For details of the FORMAT clause, see 4.2.

Simply use a single input file for multiple LOAD INDEX processes (sharing the same input file among simultaneous LOAD INDEX processes). For example, use the input file for the primary key loading for any other index loading in that table.

Note: *Use the LOAD INDEX for reloading into a specific index (and a specific partition as well), not just for the initial loading process. This will help when recovering a particular file at occasions of database/disc corruption.*

The following subsections show the typical two patterns of the parallel load by using the INDEX clause and PARTITION clause.

D.7.1. Loading into Each Index in Parallel

In this case, simultaneously load into multiple indexes using the LOAD INDEX clause. Each partition of the specified index is loaded one by one, in a sequential manner.

The following figure illustrates this loading process.

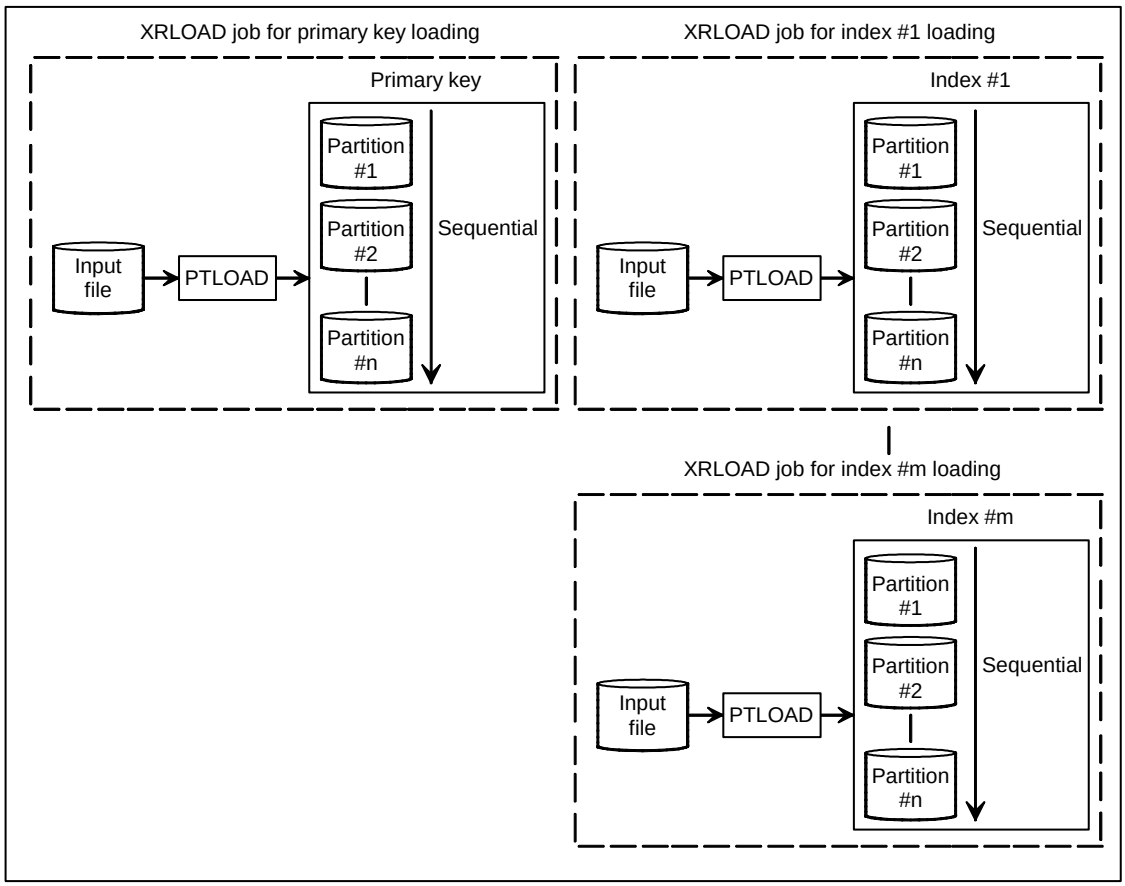


Figure D–2. Loading Process into Each Index in Parallel

Perform the following steps:

1. Prepare an input file for the primary key loading. Then, if possible, prepare input files for each secondary index and unique constraint loading.
2. Call XRLOAD and enter a LOAD command with the INDEX clause for an index.
3. XRLOAD generates a runstream for the load. The generated runstream is added automatically.
4. From the runstream, XRLOAD calls PTLOAD to load data into the index. PTLOAD loads records into one partition after another, in a sequential manner.

Step 2 can be done in parallel. For each B-tree in the table, repeat step 2. This way, you can load into multiple indexes simultaneously.

D.7.2. Loading into Each Partition in Parallel

In this case, simultaneously load into multiple partitions of an index. To do this, prepare an input file for each partition of each index, using the PARTITION clause.

The following figure illustrates this loading process.

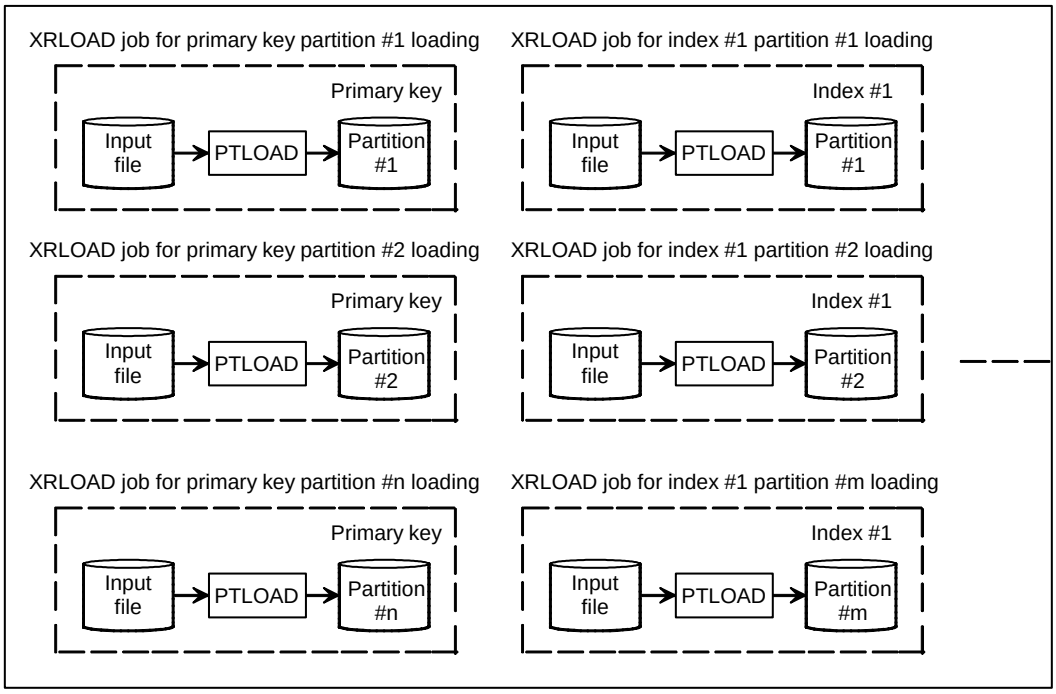


Figure D–3. Loading Process into Each Partition in Parallel

Perform the following steps:

1. Prepare an input file. Then, if possible, prepare input files for each partition.
2. Call XRLOAD and enter a LOAD command with the INDEX clause and PARTITION clause.
3. XRLOAD generates a runstream for the load. The generated runstream is added automatically.
4. From the runstream, XRLOAD calls PTLOAD to load data into the specified partition of the specified index.

Step 2 can be done in parallel. For each partition of the index, repeat step 2. This way, you can load into multiple partitions simultaneously.

Note: *If a PARTITION clause is specified, the records that are not within the specified partitions are ignored. In other words, you can use the input file for loading to the table.*

D.8. Table Definition

The table and partition definition is as follows:

```
CREATE TABLE SCH.TBL (
   COL1:DEC(22)         NOT NULL,
   COL2:DEC(22.3)       NOT NULL,
   COL3:NUMERIC(11)     NOT NULL,
   COL4:NUMERIC(11.3)   NOT NULL,
   COL5:INTEGER         NOT NULL,
   COL6:CHAR(30)         NOT NULL,
   COL7:CHAR(30)         NOT NULL,
   COL8:CHAR(35)         NOT NULL,
   COL9:DEC(22)       NOT NULL,
   COL10:DEC(22)         NOT NULL,
 CONSTRAINT PK PRIMARY KEY ( COL1,COL3,COL6,COL4,COL2 ASC ),
 CONSTRAINT UQ UNIQUE ( COL10 ASC,COL7 ASC,COL4 ASC ),
 INDEX IX1 ON COL9 ASC,COL5 ASC,COL8 ASC,COL4 ASC,COL2 ASC
 PARTITION PRIMARY KEY
   PK_PART_1 <= (10)         IN SCH.AREA_PK_1,
   PK_PART_2 <= (20)         IN SCH.AREA_PK_2,
   PK_PART_3 <= (MAXVALUE)   IN SCH.AREA_PK_3
 PARTITION INDEX UQ
   UQ_PART_1 <= (10)         IN SCH.AREA_UQ_1,
   UQ_PART_2 <= (20)         IN SCH.AREA_UQ_2,
   UQ_PART_3 <= (MAXVALUE)   IN SCH.AREA_UQ_3
 PARTITION INDEX IX1
   IX1_PART_1 <= (10)        IN SCH.AREA_IX1_1,
   IX1_PART_2 <= (20)        IN SCH.AREA_IX1_2,
   IX1_PART_3 <= (MAXVALUE)  IN SCH.AREA_IX1_3
);
```

D.9. XRLOAD Execution Runstream

The following runstream illustrates the execution of XRLOAD for loading into the partitioned table calling XRLOAD once:

```
@MSG,N  *** Load External Format(XRLOAD) ***
@SYS$LIB$*IRU.IRU,E
DOWN FILE SCH*AREA-PK-1 ;
DOWN FILE SCH*AREA-PK-2 ;
DOWN FILE SCH*AREA-PK-3 ;
DOWN FILE SCH*AREA-IX1-1 ;
DOWN FILE SCH*AREA-IX1-2 ;
DOWN FILE SCH*AREA-IX1-3 ;
DOWN FILE SCH*AREA-UQ-1 ;
DOWN FILE SCH*AREA-UQ-2 ;
DOWN FILE SCH*AREA-UQ-3 ;
ACT APPL UDSSRC ;
END ;
@MSG, N *** Detach partitions ***
```

```
@SYS$LIB$*RDMS.RDMUTL, EA   UDSSRC
DETACH PARTITION  PK_PART_1 FROM TABLE SCH.TBL ;
DETACH PARTITION  PK_PART_2 FROM TABLE SCH.TBL ;
DETACH PARTITION  PK_PART_3 FROM TABLE SCH.TBL ;
DETACH PARTITION  UQ_PART_1 FROM TABLE SCH.TBL ;
DETACH PARTITION  UQ_PART_2 FROM TABLE SCH.TBL ;
DETACH PARTITION  UQ_PART_3 FROM TABLE SCH.TBL ;
DETACH PARTITION  IX1_PART_1 FROM TABLE SCH.TBL ;
DETACH PARTITION  IX1_PART_2 FROM TABLE SCH.TBL ;
DETACH PARTITION  IX1_PART_3 FROM TABLE SCH.TBL ;
EXIT ;
@USE      XR$SORT.,SORT$FILE.
@USE      ABS.,XRUTL.
@ASG,T    XR$SELECT.,F///262143
@ASG,T    XR$SORTOUT.,F///262143
@XRUTL.XRLOAD,E  UDSSRC
LOAD FILE "INF" SORTED YES INTO SCH.TBL FORMAT
    COL1 (01:22),
    COL2 (23:45),
    COL3 (46:57),
    COL4 (58:70),
    COL5 (46:57),
    COL6 (71:100),
    COL7 (71:100),
    COL8 (71:100),
    COL9 (01:22),
    COL10 (01:22) ;
EXIT;
@SYS$LIB$*IRU.IRU,E
UP FILE SCH*AREA-PK-1 ;
UP FILE SCH*AREA-PK-2 ;
UP FILE SCH*AREA-PK-3 ;
UP FILE SCH*AREA-IX1-1 ;
UP FILE SCH*AREA-IX1-2 ;
UP FILE SCH*AREA-IX1-3 ;
UP FILE SCH*AREA-UQ-1 ;
UP FILE SCH*AREA-UQ-2 ;
UP FILE SCH*AREA-UQ-3 ;
ACT APPL UDSSRC ;
END ;
@MSG,N  *** Attach partitions ***
@SYS$LIB$*RDMS.RDMUTL,EA   UDSSRC
ATTACH PARTITION  PK_PART_1 TO TABLE SCH.TBL ;
ATTACH PARTITION  PK_PART_2 TO TABLE SCH.TBL ;
ATTACH PARTITION  PK_PART_3 TO TABLE SCH.TBL ;
ATTACH PARTITION  UQ_PART_1 TO TABLE SCH.TBL ;
ATTACH PARTITION  UQ_PART_2 TO TABLE SCH.TBL ;
ATTACH PARTITION  UQ_PART_3 TO TABLE SCH.TBL ;
ATTACH PARTITION  IX1_PART_1 TO TABLE SCH.TBL ;
ATTACH PARTITION  IX1_PART_2 TO TABLE SCH.TBL ;
ATTACH PARTITION  IX1_PART_3 TO TABLE SCH.TBL ;
EXIT ;
```

D.10. Runstream Generated by XRLOAD

The following runstream is generated by XRLOAD after it is executed:

```
@MSG    **** XRLOAD START ****
@ABS.PTLOAD UDSSRC,TPF$,$XRLOAD$,PK
@PMD,ED
@MSG    **** XRLOAD 1st Index (PK) END ( 1/ 3) ****
@XR$SORT.SORT,S
FILEIN=XR$SELECT
FILEOUT=XR$SORTOUT
SELECT=0147,0022:0083,0004:0117,0030:0057,0004:;
00000000061,0022:0001,0082
KEY=0147,0022,R,A:0083,0004,B,A:0117,0030,S,A:0057,0004,B,A:0061,0022,R,A:;
00000001,0022,R,A:0023,0004,B,A:0027,0030,S,A:0057,0004,B,A:0061,0022,R,A
@EOF
@ABS.PTLOAD UDSSRC,TPF$,$XRLOAD$,IX01
@PMD,ED
@MSG    **** XRLOAD 2nd Index (IX1) END ( 2/ 3) ****  @XR$SORT.SORT,S
FILEIN=XR$SELECT
FILEOUT=XR$SORTOUT
SELECT=0169,0022:0087,0030:0057,0004:0001,0082
KEY=0169,0022,R,A:0087,0030,S,A:0057,0004,B,A
@EOF
@ABS.PTLOAD UDSSRC,TPF$,$XRLOAD$,IX02
@PMD,ED
@MSG    **** XRLOAD 2nd Index (UQ) END ( 3/ 3) ****
@MSG    **** XRLOAD END ****
```

D.11. Execution Result of XRLOAD

The following runstream illustrates the executed result of XRLOAD:

```
@XRUTL.XRLOAD,E  UDSSRC
XRLOAD 5R2-1A01*00 PROCESSOR  04/16/01   23:23:38
Please enter LOAD COMMAND followed by a semicolon (;)
   1.  LOAD FILE "INF" SORTED YES INTO SCH.TBL FORMAT
   2.      COL1 (01:22),
   3.      COL2 (23:45),
   4.      COL3 (46:57),
   5.      COL4 (58:70),
   6.      COL5 (46:57),
   7.      COL6 (71:100),
   8.      COL7 (71:100),
   9.      COL8 (71:100),
  10.      COL9 (01:22),
  11.      COL10 (01:22) ;
                  ***** XRLOAD  END  *****
@MSG    **** XRLOAD START ****
@ABS.PTLOAD UDSSRC,TPF$,$XRLOAD$,PK
PTLOAD  start  at   04/16/01 23:23:44
Parition name                PK_PART_1
Information of storage-area  "rsm-data-index"
Schema name                  SCH
Storage-area name            AREA_PK_1
        The total data  page is      1  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      10
Parition name                PK_PART_2
Information of storage-area  "rsm-data-index"
Schema name                  SCH
Storage-area name            AREA_PK_2
        The total data  page is      1  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      10
Parition name                PK_PART_3
Information of storage-area  "rsm-data-index"
Schema name                  SCH
Storage-area name            AREA_PK_3
        The total data  page is      2  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      70
```

```
The total number of rows loaded is       90
PTLOAD    end   at  04/16/01 23:23:45
@PMD,ED
@MSG    **** XRLOAD 1ST INDEX (PK) END ( 1/ 3) ****
@XR$SORT.SORT,S
SORT  2OR2A SL76R2 04/16/01 23:23:46
FILEIN=XR$SELECT
FILEOUT=XR$SORTOUT
SELECT=0147,0022:0083,0004:0117,0030:0057,0004:;
00000000061,0022:0001,0082
KEY=0147,0022,R,A:0083,0004,B,A:0117,0030,S,A:0057,0004,B,A:0061,0022,R,A :;
00000001,0022,R,A:0023,0004,B,A:0027,0030,S,A:0057,0004,B,A:0061,0022,R,A

          90 RECORDS IN XR$SELECT

*** SORT/MERGE SUMMARY ***

OPERATION = SORT

INPUT STATISTICS:
          90 RECORDS IN            0 SKIPPED                    0 REJECTED
        2047 WORDS/RECORD ON INPUT
          90 RECORDS SORTED
          82 WORDS/RECORD THROUGH SORT

OUTPUT STATISTICS:
          90 RECORDS OUT           0 DELETED ON OUTPUT
          41 WORDS/RECORD ON OUTPUT
        8960 WORDS/BLOCK ON OUTPUT
@ABS.PTLOAD UDSSRC,TPF$,$XRLOAD$,IXO1
PTLOAD  start  at   04/16/01 23:23:46
Parition name                IX1_PART_1
Information of storage-area  "rsm-index-only"
Schema name
Storage-area name            AREA_IX1_1
        The total data  page is      1  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      10
Parition name                IX1_PART_2
Information of storage-area  "rsm-index-only"
Schema name
Storage-area name            AREA_IX1_2
        The total data  page is      1  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      10
```

```
Parition name                 IX1_PART_3
Information of storage-area  "rsm-index-only"
Schema name
Storage-area name             AREA_IX1_3
        The total data  page is      2  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      70
The total number of rows loaded is       90
PTLOAD    end   at  04/16/01 23:23:47
@PMD,ED
@MSG    **** XRLOAD 2ND INDEX (IX1) END ( 2/ 3) ****
@XR$SORT.SORT,S
SORT  20R2A SL76R2 04/16/01 23:23:48
FILEIN=XR$SELECT
FILEOUT=XR$SORTOUT
SELECT=0169,0022:0087,0030:0057,0004:0001,0082
KEY=0169,0022,R,A:0087,0030,S,A:0057,0004,B,A

          90 RECORDS IN XR$SELECT

*** SORT/MERGE SUMMARY ***

OPERATION = SORT

INPUT STATISTICS:
          90 RECORDS IN            0 SKIPPED                     0 REJECTED
        2047 WORDS/RECORD ON INPUT
          90 RECORDS SORTED
          49 WORDS/RECORD THROUGH SORT

OUTPUT STATISTICS:
          90 RECORDS OUT           0 DELETED ON OUTPUT
          35 WORDS/RECORD ON OUTPUT
        8960 WORDS/BLOCK ON OUTPUT
@ABS.PTLOAD UDSSRC,TPF$,$XRLOAD$,IX02
PTLOAD  start  at   04/16/01 23:23:48
Parition name                 UQ_PART_1
Information of storage-area  "rsm-index-only"
Schema name
Storage-area name             AREA_UQ_1
        The total data  page is      1  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      10
Parition name                 UQ_PART_2
Information of storage-area  "rsm-index-only"
```

```
Schema name
Storage-area name             AREA_UQ_2
        The total data  page is      1  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      10
Parition name                 UQ_PART_3
Information of storage-area  "rsm-index-only"
Schema name
Storage-area name             AREA_UQ_3
        The total data  page is      2  on this B-tree
        The total index page is      1  on this B-tree
        The last page number is     13  on this B-tree
This B-tree level is       2
The number of rows loaded for this partition is      70
The total number of rows loaded is       90
PTLOAD    end   at  04/16/01 23:23:49
@PMD,ED
@MSG    **** XRLOAD 2ND INDEX (UQ) END ( 3/ 3) ****
@MSG    **** XRLOAD END ****
```

Appendix E
Relational Database Fast Load Error Messages

This appendix describes XRLOAD/ JETLOAD/PTLOAD error messages, explains message categories, and includes examples.

Messages are listed in numerical order and include the following information:

- Message number
- Message type identifiers:
  - Display type
  - Dump type
  - Module type
- Message text
- Any relevant comments or suggestions

E.1. Error Message Description

Each XRLOAD error message consists of a message number followed by the message text. The message number identifies both the message and its range or category. A column name is displayed if XRLOAD can identify the column that caused the error. For syntax errors, XRLOAD places an asterisk (*) directly below the item that caused the error.

Snap dumps with diagnostic information appear following the error message for some logical, facility, I/O, and fatal errors:

Snap-id	Description
AIT	Data or index area information
AIT-DA	Data area information
AIT-IN	Index area information
ALLOC	Contents of allocation page
ASGIMG	File assignment image
BCB	PCIOS Buffer Control Block
CNVPKT*	Data conversion packet (the upper half of the third snapped word indicates the item code of the error column; the upper half of the fourth word indicates the item size in bytes after conversion)
CSFIMG	ECL image processed by CSF$
CSFSTS	Status returned by CSF$
ELTRWP	Packet for reading omnibus element
ERRCHR*	Error column number in input record
FCB	PCIOS file control block
FCLPKT	FACIL$ or DM$FAC packet
FCP	Contents of file control page
FCT	PCIOS file control table
INRBSZ*	Actual record size in bytes of input file
IOBUF	I/O buffer for IO$ or IOW$
IOBUF1	I/O buffer for reading from or writing to SDF file
IOBUF2	I/O buffer for reading from or writing to SDF file
IOPKT	I/O packet
ITMCNT*	Item code of column JETLOAD or PTLOAD was processing
JCT	Control table used by JETLOAD or PTLOAD
NEWREC*	Image of record JETLOAD or PTLOAD was processing on error, in RDMS data record format (only key values guaranteed; key compression not performed)
OLDREC*	Image of record JETLOAD or PTLOAD just processed, in RDMS data record format
OMNIMG	Table-id and length of omnibus element
OPTOPR	Information on options and operands on processor call statement
PWK1-2	PCIOS work area

Snap-id	Description
RECCNT*	Line number at which JETLOAD or PTLOAD finished processing
RECIMG*	Image of input record JETLOAD or PTLOAD was processing
RINFOR	Contents of INFOR table
RINFPK	Status returned by RINF$ and address of INFOR table
SCT	PCIOS storage control block
SDFIWK	Work area for reading from or writing to SDF file
SDFPKT	Packet for reading from or writing to SDF file
WRTPKT	Packet for writing to SDF file
WT-PKT	Packet for reading from SDF file
XRRBSZ*	Expected record size in bytes of input file

* Most useful reason for error.

Immediately following the message number in this appendix is a 3-part type identifier, as in the following example:

```
10001:M:N:X    The name of the application group is missing.
```

In this example, 10001 is the message number; M, N, and X are its type identifiers.

Note: *Type identifiers are not displayed with the message. They are included only in this appendix.*

The type identifiers are as follows:

- The display type indicates what is displayed:
  - M Only the error message
  - P The error message and error location in the LOAD command
  - C The error message and error column name
- The dump type indicates whether or not a postmortem dump (PMD) was produced when the error occurred:
  - Y A PMD was produced
  - N No dump was produced (Produce a dump by specifying the P option on the processor call statement.)
- The module type indicates the module in which the message occurred.
  - X XRLOAD
  - J JETLOAD or PTLOAD
  - O XRLOAD or JETLOAD or PTLOAD

E.2. Error Categories

XRLOAD error messages fall into nine categories:

Type of Error	Error Number Range
Initial	10000 to 19999
Syntax	20000 to 29999
Definition	30000 to 39999
Generation	40000 to 49999
Logical	50000 to 59999
Facility	60000 to 69999
I/O	70000 to 79999
Contingency	80000 to 89999
Fatal	90000 to 99999

Examples

The following example includes a syntax error message:

```
@XRUTL.XRLOAD,E UDSSRC
XRLOAD 4R1-1A01*00 PROCESSOR  10/28/97   13:47:02
Please enter LOAD COMMAND followed by a semicolon (;)
   1.  LOAD INTERNAL FILE "DATAF" SORTED YES
   2.  INTO FT5007.TBL1:A FORMAT
INTO FT5007.TBL1:A FORMAT
                       *
*ERROR* [ 20104 ]
        The keyword FORMAT is not allowed for INTERNAL or USER format
        input files.
```

The following example includes a column name error message:

```
@XRUTL.XRLOAD,E UDSSRC
XRLOAD 4R1-1A01*00 PROCESSOR  10/28/97   15:25:35
Please enter LOAD COMMAND followed by a semicolon (;)
   1.  LOAD EXTERNAL FILE "F" SORTED YES
   2.  INTO FT5030.TBL1:A FORMAT
   3.  COL01 ( 1),
   4.  COL02 (14:20)   NULLIF ((14:20) = '+123456'),
   5.  NGCOL (54),
   6.  COL04 (59:62),
*ERROR COLUMN IS        NGCOL
*ERROR* [ 30041 ]
        Nonexistent column-name specified.
```

The final example includes a logical error message with snap dumps. It illustrates that the error occurred for column COL1 at column 3 of line 2 in the input file.

The table definition is as follows:

```
CREATE TABLE FT4016.TBL IN FT4016.NAREA
  COLUMNS ARE COL1:SMALLINT,
              COL2:SMALLINT
  PRIMARY KEY PK IS COL1 ASC ;
```

The input file contains the following data:

```
+131070 +131070
 1+1071 +131071
+131072 +131072
```

XRLOAD returns the following lines:

```
@ABS.JETLOAD UDSSRC,OMN-FILE,$XRLOAD$,PK
JETLOAD  start  at  08/05/98 09:34:49
*ERROR* [50006]
Invalid character
RECCNT      151633            042001                  09:34:49
08/05/98          1100 TIME-SHARING EXEC

151560                                  000000000001
ERRCHR      151633            042001                  09:34:49
08/05/98          1100 TIME-SHARING EXEC

151560                                                  000000000003
CNVPKT      151633            042001                  09:34:49
08/05/98          1100 TIME-SHARING EXEC

121130                                  153354000000   16716500020
121140   000001000000   000002000000
RECIMG      151633            042001                  09:34:49
08/05/98          1100 TIME-SHARING EXEC

153350     040061053061   060067061040   053061063061   060067061040
```

E.3. Initial Errors

10001:M:N:X **The name of the application group is missing.**

Enter an application group name.

10002:M:N:X **The application group name cannot exceed 6 characters.**

Reenter the name of the application group in six characters or less.

10004:M:N:X **The save file must be attached by a USE name to the file.**

The @USE command was not executed on the runstream save file before XRLOAD was called.

10005:M:N:X **The XR$SORT file must be attached by a USE name to the file.**

The @USE command was not executed on the sort processor file as XR$SORT before XRLOAD was called.

10006:M:N:X **The ABS file must be attached by a USE name to the file.**

The @USE command was not executed on the JETLOAD processor file as ABS before XRLOAD was called.

10007:M:N:X **XRLOAD cannot assign the XR$SELECT file.**

The file for which the @USE command was executed as XR$SELECT before XRLOAD was called may not be cataloged; or another user may have assigned the file exclusively.

10008:M:N:X **XRLOAD cannot assign the XR$SORTOUT file.**

The file for which the @USE command was executed as XR$SORTOUT before XRLOAD was called may not be cataloged; or another user may have assigned the file exclusively; or XRLOAD may have failed to assign the file with the T option.

10011:M:N:X **XRLOAD cannot assign the input record file.**

The input file may not be cataloged; or another user may have assigned the file exclusively; or the @USE command was not executed on the input file as a user-specified name before XRLOAD was called.

10012:M:N:X **Invalid option specified on the processor call statement. Valid options are C,E,K,L,N,P and Z.**

The only options allowed on the processor call statement are C, E, K, L, N, P and Z.

10013:M:N:X **Invalid parameter is specified on the processor call statement.**

Reenter the parameters on the processor call statement in the format *application-group-name*[,*use-name-2*.[*element-name*]].

10014:M:N:X **The save file cannot be assigned to the tape file.**

You must use the disk file as the file where XRLOAD stores the generated runstream for a LOAD command.

10015:M:N:X **The ABS file cannot be assigned to the tape file.**

You must use the disk file as the file where XRLOAD expects its storage manager JETLOAD to reside.

10016:M:N:X **The XR$SORT file cannot be assigned to the tape file.**

You must use the disk file as the file where XRLOAD expects the SORT processor to reside.

10017:M:N:X **The XR$SELECT file cannot be assigned to the tape file.**

You must use the disk file as the file where JETLOAD places items used for building secondary index and unique constraint B-trees or for loading from an EXTERNAL format input file that is not sorted in the table's primary key order.

10018:M:N:X **The XR$SORTOUT file cannot be assigned to the tape file.**

You must use the disk file as the file where the SORT processor places the sort result.

E.4. Syntax Errors

20001:P:N:X **The keyword EXTERNAL, INTERNAL, or USER is out of sequence.**

The keyword EXTERNAL, INTERNAL, or USER is out of sequence on the LOAD command.

20004:P:N:X **The load factor must be between 50 and 100.**

Change the load factor to a value between 50 and 100.

20005:P:N:X **A character string exceeds 30 characters.**

A string of characters cannot exceed 30 characters.

20007:P:N:X **The LOAD command contains an invalid character.**

20008:P:N:X **The internal file name is too long. It cannot exceed 12 characters.**

Reenter the internal file name in 12 characters or less.

20010:P:N:X **The command line exceeds 80 characters.**

A command line cannot exceed 80 characters.

20011:P:N:X **The start-column position is higher than the end-column position.**

The start-column position must have a lower number than the end-column position in the item-position clause.

20015:P:N:X **An illegal table name was specified.**

Reenter the table name in the following format:

```
[qualifier.]table-name[:version-name].
```

20016:P:N:X **Missing null character string for NULLIF clause.**

Specify a string of characters for the NULLIF clause; or remove the NULLIF clause.

20018:P:N:X **The position specified in the NULLIF clause must fall within the range specified in the item-position field.**

The NULLIF clause must have a starting and ending position for null characters, which must fall within the starting and ending position for the corresponding column.

20019:P:N:X **The XRLOAD processor terminated before the statement ended.**

Ensure that an end-of-file marker does not precede the semicolon.

20020:P:N:X **The CORE parameter contains an invalid value.**

The CORE parameter accepts only numerical values or K and M.

20021:P:N:X **The keyword SORTED is out of sequence.**

The keyword SORTED must follow the input file name.

20022:P:N:X **The keyword YES is out of sequence.**

The keyword YES must follow the keyword SORTED.

20023:P:N:X **The keyword CORE is out of sequence.**

The keyword CORE must follow the input file name or SORTED clause.

20024:P:N:X **The keyword FACTOR is out of sequence.**

The keyword FACTOR must follow the keyword LOAD.

20025:P:N:X **The keyword INTO is out of sequence.**

The INTO clause must be the last clause when loading an INTERNAL or USER format file, or it must precede the FORMAT clause for an EXTERNAL format file.

20026:P:N:X **The keyword FORMAT is out of sequence.**

The keyword FORMAT must immediately follow the table specification or the INDEX clause.

20027:P:N:X **The keyword NULLIF is out of sequence.**

The keyword NULLIF must follow the start-column and end-column, if used, in the item-position clause.

20028:P:N:X **The keyword POSITION is out of sequence.**

The keyword POSITION can be used for an INTERNAL or a USER format file, and if used, it must follow the column-name in the item-position field.

20030:C:N:X **XRLOAD cannot load approximate numeric data from EXTERNAL format files.**

Approximate numeric data is not allowed in EXTERNAL format files.

20031:C:N:X **The size of NCHARACTER columns is not even.**

The size of NCHARACTER columns must be an even number.

20032:C:N:X **XRLOAD cannot load approximate numeric data into key columns.**

Approximate numeric data is not allowed in primary key, secondary index, or unique key columns.

20035:P:N:X **The file-name is out of sequence.**

The input-file-name must immediately follow the keyword FILE.

20036:C:N:X **The number of characters specified in the NULLIF clause exceeds the specified column length.**

The number of null characters cannot exceed the assumed length calculated from the starting and ending position in the NULLIF clause.

20040:P:N:X **The CORE parameter exceeds 8 digits.**

Reenter the CORE parameter in 8 digits or less.

20042:P:N:X **The specified symbolic character is out of sequence.**

A symbolic character (such as , or =) is out of sequence.

20043:P:N:X **The LOAD command contains an invalid keyword or symbolic character.**

20044:P:N:X **The size of the item-position field is greater than SDF MAX length of 30000.**

The value for the position of the column or the null indicator pattern cannot exceed the maximum SDF record size of 8,188 characters.

20045:P:N:X **The specified number is out of sequence.**

20046:P:N:X **The specified literal is out of sequence.**

20048:C:N:X **Invalid null character string on NULLIF clause for BLOB columns. For BLOB columns, the character string field must not start with a number.**

Null character string on NULLIF clause must start with an alphabetical character or special character.

20049:P:N:X **The keyword NO is out of sequence.**

The keyword NO must follow the keyword SORTED.

20050:P:N:X **A pair of double quotes is needed.**

If you specified an external file name, you must enclose it in double quotation marks; or if you started from a double quotation mark, you must end with a double quotation mark.

20051:P:N:X **A double quote is out of sequence.**

A double quotation mark must be used to enclose the input file name.

20052:P:N:X **The file name is too long. It cannot exceed 12 characters.**

Reenter the file name in 12 characters or less.

20053:P:N:X **The file cycle is too long. It cannot exceed 3 characters.**

Reenter the file cycle in 3 characters or less.

20054:P:N:X **The read/write key is too long. It cannot exceed 6 characters.**

Reenter the read or write key in 6 characters or less.

20055:P:N:X **The external file name is too long. It cannot exceed 30 characters.**

Use the @USE command to attach the internal file name to the external file name and reenter the internal file name instead of the external file name.

20056:P:N:X **The input file name contains an invalid character.**

20057:P:N:X **The keyword TABLE is out of sequence.**

The keyword TABLE, if used, must follow the keyword INTO.

20058:P:N:X **The keyword INDEX is out of sequence.**

The INDEX clause must immediately follow the table specification.

20059:P:N:X **The keyword PARTITION is out of sequence.**

The keyword PARTITION must immediately follow the index name.

20060:P:N:X **A number of specified partition-names exceed the limit**.

The number of specified partition-names must be 511 or less.

20101:P:N:X **The keyword LOAD is out of sequence.**

The keyword LOAD must be the first word in the LOAD command.

20103:P:N:X **The keyword FILE is out of sequence.**

The keyword FILE must immediately precede the input-file name.

20104:P:N:X **The keyword FORMAT is not allowed for INTERNAL or USER format input files.**

Data does not have to be formatted for INTERNAL or USER format input files except when the INDEX clause is specified. Thus, the keyword INTERNAL or USER is invalid.

20107:P:N:X **The number of null characters specified is too long.**

Decrease the number of null characters specified in the NULLIF clause.

20501:M:N:X **XRLOAD cannot use the external input file.**

Correct the external input file name specification.

E.5. Errors 30000 to 99999

Errors 30000 to 99999 can sometimes be difficult to resolve**.** Consult with your database administrator, systems analyst, or support personnel if you are unable to resolve an error.

E.6. Definition Errors

30001:M:Y:X **An ADT entry is missing in file UDS$$SRC*ADT-FILE.**

30002:M:Y:X **XRLOAD cannot assign file UDS$$SRC*ADT-FILE.**

30003:M:Y:X **XRLOAD cannot read file UDS$$SRC*ADT-FILE.**

30005:M:Y:X **XRLOAD cannot access file UDS$$SRC*ADT-FILE.**

The internal @USE command attached to file UDS$$SRC*ADT-FILE failed.

30006:M:Y:X **XRLOAD cannot free file UDS$$SRC*ADT-FILE.**

30011:M:Y:X **XRLOAD cannot assign file SYS-FILE.**

30012:M:Y:X **XRLOAD cannot read file SYS-FILE.**

30013:M:Y:X **XRLOAD cannot read DOWN LIST.**

30014:M:Y:X **XRLOAD cannot access file SYS-FILE.**

The internal @USE command attached to file SYS-FILE failed.

30015:M:Y:X **XRLOAD cannot free file SYS-FILE.**

30016:M:Y:X **XRLOAD cannot handle this file SYS-FILE.**

The file SYS-FILE is not in the format XRLOAD can handle.

30021:M:Y:X **XRLOAD cannot assign file FDT$.**

30022:M:Y:O **XRLOAD cannot read file FDT.**

30023:M:Y:O **XRLOAD cannot find an FDT entry for the storage area to load.**

Ensure that a file description table (FDT) entry exists for the storage area.

30031:M:Y:X **XRLOAD cannot assign file RDT$FILE.**

The DOMAIN attribute may be set to UDS**.** It should be USER-UDS.

30032:M:Y:X **XRLOAD cannot read file RDT$FILE.**

30033:M:Y:X **XRLOAD cannot find the table information in file RDT$FILE.**

Ensure that the table you are attempting to load is recorded in file RDT$FILE**.** The table may not exist and must be created.

30034:M:Y:X **XRLOAD cannot find an RDT entry in file RDT$FILE.**

Ensure that an RDT entry exists for the table.

30035:M:Y:X **XRLOAD cannot find an FDT entry for file RDT$FILE in file FDT$.**

Ensure that an FDT entry for file RDT$FILE exists in file FDT$.

30041:C:N:X **Nonexistent column-name specified.**

The column-name specified does not exist for the table definition.

30042:C:N:X **A NULLIF clause is specified for a NULL NOT ALLOWED column.**

The NULLIF clause cannot be specified for a NULL NOT ALLOWED column.

30043:C:N:X **The size of the column is in error.**

Decrease the size of the column within the maximum input data size of XRLOAD.

30044:M:N:X **The number of items specified in the item-position field does not correspond with the number of columns in the table.**

The number of columns specified does not correspond with the number of columns in the table definition when loading without the INDEX clause. If the INDEX clause is specified, some of the index columns in the table definition are not specified.

30045:C:N:X **The size of the item-position field is greater than SDF MAX length of 8188.**

Decrease the size of the item-position field.

30046:M:Y:X **The item-data-type in the RDT is invalid.**

A nondefault character set or collation cannot be specified in the column definition.

30047:C:N:X **The column name specified on the NULLIF clause is not specified in the FORMAT clause of the LOAD command.**

The column name specified on the NULLIF clause must be specified in the FORMAT clause of the LOAD command.

30052:M:Y:X **XRLOAD cannot assign file PART$FILE.**

30053:M:Y:X **XRLOAD cannot read file PART$FILE.**

30055:M:Y:X **XRLOAD cannot find an FDT entry for file PART$FILE in file FDT$.**

Ensure that an FDT entry for file RDMS.RDMSPARTITIONS exists in file FDT$.

30056:M:Y:X **PART$FILE is contradictory to RDT$.**

The information in RDMS.RDMSPARTITIONS is inconsistent with that in file RDT$.

30060:M:N:X **Invalid partition-name is specified.**

There are either nonregistered partition-names or duplicate partition-names.

30061:C:N:X **XRLOAD cannot load NULL ALLOWED column for primary or specified index key from INTERNAL or USER format file that is not sorted in that primary key order.**

Primary keys cannot contain NULL ALLOWED columns if you use an INTERNAL or a USER format input file that is not sorted in the table's primary key order when loading without the INDEX clause. If the INDEX clause is specified, the specified index keys the same restriction.

30062:P:N:X **Nonexistent index-name specified.**

The index-name specified does not exist for the table definition.

30063:P:N:X **The INDEX clause is allowed only for the partitioned table.**

The INDEX clause cannot be specified for the table that is not partitioned.

30091:M:Y:X **XRLOAD cannot release file SYS-FILE.**

XRLOAD is unable to free file SYS-FILE internally.

30092:M:Y:O **The status of the file is not DOWN.**

Execute the IRU DOWN command on the storage area into which you plan to load.

30093:M:Y:J **The partition-state of the partition is neither "Detach" or "Hidden".**

Execute RDMUTL DETACH PARTITION or HIDE PARTITION command on the partition where data is loaded.

30094:M:Y:X **SORT library error in open process.**

30095:M:Y:X **SORT library error in release process.**

30096:M:Y:X **SORT library error in sort process.**

30097:M:Y:X **SORT library error in retrieve process.**

E.7. Generation Errors

40001:M:Y:X **No file is assigned for the generated runstream.**

40002:M:Y:X **An error status was returned during the call to SOR-OUT.**

40003:M:Y:X **An error status was returned during the call to SOR-FINL.**

40010:M:Y:X **XRLOAD cannot write to omnibus element for load definition.**

40011:M:Y:X **XRLOAD cannot write to omnibus element for RDT definition.**

40012:M:Y:X **XRLOAD cannot write to omnibus element for FDT definition.**

40013:M:Y:X **The file into which to place the generated JCL is too small. You should expand internal file *file-name.***

file-name is the name of the current internal file. This may be an I/O error 022. Ensure that the file has sufficient space to receive the data.

40014:M:Y:X **XRLOAD cannot write to omnibus element for down list definition.**

40015:M:Y:X **XRLOAD cannot write to omnibus element for PARTITION_PAGE definition.**

40016:M:Y:X **XRLOAD cannot write to omnibus element for PARTITION_INFO definition.**

40017:M:Y:X **Input file count is exceeded. Five inputs are the maximum.**

40018:M:Y:X **Tapes cannot be used as multiple file inputs.**

E.8. Logical Errors

50001:M:Y:J **Key sequence error.**

The input file is not sorted by primary key.

50002:M:Y:J **Key duplication error.**

Primary key values are duplicated on this B-tree.

50006:M:Y:J **Invalid character.**

Numeric data contains alphabetic characters without a decimal point, sign, and leading space.

Dateline data contains alphabetic characters without a decimal point, hyphen, and colon.

50007:M:Y:J **Sign duplication.**

Numeric data has more than one sign.

50009:M:Y:J **Binary item bit overflow.**

The bit size is overflowing the column definition. XRLOAD cannot store this value.

50010:M:Y:J **There is no input data.**

The input file has no apparent records. It may be that the input file was rewritten during the execution of XRLOAD.

50011:M:Y:J **Index level overflow.**

The B-tree level exceeds seven. The limit is seven. Increase the PAGE-SIZE attribute.

50012:M:Y:J **Allocated pages or file catalog size overflow.**

The page number is overflowing on either the storage area definition or file catalog size. Increase the MAXIMUM-PAGES attribute or file catalog size.

50013:M:Y:J **FCP format-id error.**

The FORMAT-ID on the file control page (FCP) contains an invalid value.

50014:M:Y:J **File mapping between FCP and FDT does not match.**

A bad reference exists between the FCP and FDT. The database may be corrupted.

50015:M:Y:J **FCP page size error.**

The page size on the FCP contains an invalid value.

50016:M:Y:J **FCP file size error.**

The file size on the FCP contains an invalid value.

50017:M:Y:J **FCP number of allocation page error.**

The number of allocation pages on the FCP is invalid.

50018:M:Y:J **FCP update concurrencies error.**

The value of UPDATE-CONCURRENCIES on the FCP is invalid.

50019:M:Y:J **FCP MAX-SRM error.**

The value of MAX-SRM on the FCP is invalid. The relation code cannot exceed 90.

50020:M:Y:J **Allocation validation error.**

The value of VALIDATION on an allocation page is invalid.

50021:M:Y:J **Allocation page-format error.**

The page format of an allocation page is invalid.

50023:M:Y:J **Page size is too small for table size.**

A page should contain at least four records. Increase the PAGE-SIZE attribute.

50024:M:Y:J **A null indicator in the input file is invalid.**

Check the null indicators in the input file.

50025:M:Y:O **Input file record size is smaller than expected size.**

The record size of the input file is smaller than the size XRLOAD expects. The expected size is the size (WSZ) in words using the following formula:

For EXTERNAL format with FORMAT clause

WSZ = (A + 3) // 4

where A is the maximum ending position for the column in the item-position field.

For EXTERNAL format without FORMAT clause

WSZ = (B + (CÄ1) + 3) // 4

where:

B is the sum of the characters (bytes) in the columns of the table. This column size is based on the rules of corresponding data types. For more details, see 3.3.

C is the number of columns in the table.

For INTERNAL format

WSZ = (D + E + 3) // 4

For USER format

WSZ = (D + 3) // 4

where:

D is the sum of the characters (bytes) in the columns of the table. This column size is based on the presentation in the database used by RDMS. For more details, see the *Relational Database Server for ClearPath OS 2200 SQL Programming Reference Manual*.

50026:M:Y:X **END DESCRIPTION is missing in input file.**

The input file has no >END DESCRIPTION< image to mark the last line of the descriptive text.

50027:M:Y:J **The value of YEAR, MONTH, DAY, HOUR, MINUTE, or SECOND is not valid.**

The dateline value is out of range. For the valid range, see the *Relational Database Server for ClearPath OS 2200 SQL Programming Reference Manual*.

50028:M:N:X **There is no input data for partitioned tables.**

When loading into a partitioned table, an empty input file is not allowed.

50029:M:Y:X **XRLOAD cannot manage the information of the partition key definition.**

The data size of the partition key information exceeds the limit in XRLOAD. For more information, see Appendix D.

50030:M:Y:J **It is not within the range of any partition.**

There is no partition into which the partition key value of an input record is stored.

50031:M:N:X **The partition key contains both ascending and descending key columns.**

XRLOAD cannot load the table that has multiple columns partition key containing both key sequence types, ascending and descending.

50032:M:Y:J **XRLOAD cannot load internal lob-id or external lob-id for BLOB columns.**

The type of LOB identifier shows internal format LOB identifier or external format LOB identifier. XRLOAD cannot load these types of LOB identifiers. Ensure that the RDMS UNLOAD statement with WITHOUT BLOB VALUES clause creates the LOB identifier.

50033:M:Y:X **The value of lob-id for BLOB columns is invalid.**

The value of the type of LOB identifier is invalid.

E.9. Facility Errors (ER CSF$ and ER FACIL$ Errors)

60003:M:Y:J **Storage area assign error.**

Error returned by CSF$. Neither JETLOAD nor PTLOAD can assign the storage area.

60004:M:Y:J **Storage area TIP assign error.**

The storage area into which you plan to load has not been registered to TIP.

60005:M:Y:J **Input file assign error.**

Error returned by CSF$. Neither JETLOAD nor PTLOAD can assign the input file.

60006:M:Y:J **Sort output file assign error.**

60009:M:Y:J **User-specified file assign error.**

60010:M:Y:J **Input file device error.**

The input file is an invalid device type. XRLOAD cannot load this device.

60011:M:Y:J **FDT$ file assign error.**

Error returned by CSF$. PTLOAD cannot assign the FDT$ file.

E.10. I/O Errors

70001:M:Y:O **Input file read error.**

An I/O error was returned while attempting to read the input file.

70002:M:Y:J **Sort input file write error.**

An I/O error was returned while attempting to write to file XR$SELECT.

70003:M:Y:J **Storage area write error.**

An I/O error was returned while attempting to write to the storage area.

70004:M:Y:O **Input file close error.**

Error was returned from file manager when JETLOAD requested the CLOSE command to the input file.

70005:M:Y:J **Sort input file close error.**

Error was returned from file manager when JETLOAD requested the CLOSE command to file XR$SELECT.

70006:M:Y:J **Omnibus element ($XRLOAD/RDT) read error.**

An I/O error was returned while attempting to read the relation description table (RDT) information element.

70007:M:Y:J **Omnibus element ($XRLOAD/LOAD) read error.**

An I/O error was returned while attempting to read the LOAD COMMAND information element.

70008:M:Y:J **Omnibus element ($XRLOAD/FDT) read error.**

An I/O error was returned while attempting to read the FDT information element.

70009:M:Y:J **FCP read error.**

An I/O error was returned while attempting to read the FCP.

70010:M:Y:O **Input file open error.**

Error was returned from file manager when JETLOAD requested the OPEN command to the input file.

70011:M:Y:J **Sort input file open error.**

Error was returned from file manager when JETLOAD requested the OPEN command to file XR$SELECT.

70012:M:Y:J **Allocation page write error.**

An I/O error was returned while attempting to write to an allocation page.

70013:M:Y:J **FCP write error.**

An I/O error was returned while attempting to write to the FCP.

70014:M:Y:J **Allocation page read error.**

An I/O error was returned while attempting to read an allocation page.

70015:M:Y:O **Input tape format error.**

The input tape file is not a system data format (SDF) file XRLOAD can handle.

70016:M:Y:O **Input tape has invalid block size.**

70017:M:Y:O **Input tape is invalid device type.**

70018:M:Y:J **Omnibus element ($XRLOAD/DWLIST) read error.**

An I/O error was returned while attempting to read the down list information element.

70019:M:Y:J **Partition page omnibus element read error.**

An I/O error was returned while attempting to read the partition page element.

70020:M:Y:J **Partition information omnibus element read error.**

An I/O error was returned while attempting to read the partition information element.

E.11. Contingency Error

XRLOAD produces one contingency error message:

80001:M:Y:J **JETLOAD CONTINGENCY ERROR.**

JETLOAD has aborted.

E.12. Fatal Errors (JETLOAD or PTLOAD Internal Errors)

90001:M:Y:J **Invalid data item.**

XRLOAD rejected an invalid RDMS data type.

90004:M:Y:J **Operand read error.**

JETLOAD cannot read input parameter from XRLOAD.

90005:M:Y:J **Operand check error.**

JETLOAD detected an error on input parameter from XRLOAD.

90006:M:Y:J **Table-id of $XRLOAD/RDT is not RD.**

The header of the RDT information element is not equal to RD.

90007:M:Y:J **$XRLOAD/RDT size error.**

The RDT information element size is too large to read JETLOAD.

90008:M:Y:J **Table-id of $XRLOAD/LOAD is not LD.**

The header of the LOAD COMMAND information element is not equal to LD.

90009:M:Y:J **$XRLOAD/LOAD size error.**

The size of the LOAD COMMAND information element is too large to read JETLOAD.

90010:M:Y:J **Table-id of $XRLOAD/FDT is not FD.**

The header of the FDT information element is not equal to FD.

90011:M:Y:J **$XRLOAD/FDT size error.**

The size of the FDT information element is too large to read JETLOAD.

90012:M:Y:J **FCP file mapping contains an invalid value.**

Ensure that the storage area was initialized correctly.

90013:M:Y:J **Table-id of $XRLOAD/DWLIST is not DW.**

The header of the down list information element is not equal to DW.

90014:M:Y:J **$XRLOAD/DWLIST size error.**

The down list information element size is too large to read.

90015:M:Y:J **Table-id of partition page omnibus element is not PP.**

The header of the down list information element is not equal to PP.

90016:M:Y:J **Partition page omnibus element size error.**

The partition page element size is too large to read.

90017:M:Y:J **Table-id of is partition information omnibus element not PI.**

The header of the partition information omnibus element is not equal to PI.

90018:M:Y:J **Partition information omnibus element size error.**

The partition information element size is too large to read.

90019:M:N:X **Invalid partition status.**

XRLOAD rejects an invalid RDMS partition status.

Index

A

advantages over RDMUTL, 1-4
application group for loaded table, 4-3
approximate numeric data, 3-3

B

benefits compared with RDMUTL, 1-4
bypassing key compression, 4-3

C

cache hit ratio undisturbed, 1-4
calling XRLOAD
  description of lines, 4-2
  example, 4-3
  model, 4-1
character data type
  rules
    general, 3-4
CHARACTER data type
  for internal files, 3-8, A-1
CHECKSUM YES ignored, 2-4
clauses not available, 1-3
COBOL
  ASCII COBOL data types compared to RDMS column definitions, 3-8
  DATE data type assignment, 3-8, 3-9
  files in PCIOS format, 3-1
  TIME data type assignment, 3-8, 3-10
  TIMESTAMP data type assignment, 3-8, 3-10, 3-11
  UCS COBOL data types compared to RDMS column definitions, A-1
contingency error messages, E-20
CORE parameter, 4-6
CREATE STORAGE AREA command, UREP, 2-4

D

database
  constructing large in short time, 1-4
  corruption, preventing, 2-5
  dumping as precaution, 2-6
  emptying, 2-6
  inaccessibility, 2-6
  incomplete updates, 2-5
  RDMS value representations, 3-7
  reorganizing, 1-4
  restoring, 2-6
  SDF files as input, 3-1
  with many secondary indexes, 1-3
DATE data type, for internal files, 3-8, 3-9, A-4
datetime data
  rules, 3-5
DD processor command
  CREATE STORAGE AREA, 2-4
  emptying database, 2-6
  PROCESS STORAGE AREA, 2-4
  PROCESS STORAGE AREA...DELETE, 2-6
  PROCESS STORAGE AREA...INSTALL, 2-6
DECIMAL data type, for internal files, 3-8, A-2
descriptive text, 3-2
disabling corrupted storage areas, 2-6
DOMAIN attribute, changing, 2-2
DOUBLE PRECISION data type, for internal files, 3-8, 3-9, A-4
dumps on error messages, E-2

E

elapsed time, reducing, 1-4
empty input file, C-1
  examples, C-1
empty input files
  XRLOAD process, C-1
encrypted files, 2-4
error messages
  categories, E-4
  contingency, E-20

description, E-1
dumps accompanying, E-2
examples, E-4
errors
detection and termination, 5-1
during loading, 2-6
messages, lack of for lost data, 2-5
overflow, 3-4
rerunning XRLOAD on return of, 1-4
termination of loading on, 2-6
exact numeric data, 3-3
execution of XRLOAD runstream, B-2, C-2
execution result of XRLOAD, B-3, C-3
Extended Processing Complex (XPC) cache hit ratio, 1-4
external files
description of, 3-3
input file, 3-6
loading, 3-6
preparing, 3-2
RDMS datetime data representations, 3-5

F

FDT$ file
availability, 2-2
contents, 5-1
files
disk, 3-1
DOMAIN UDS, changing from, 2-2
DOMAIN USER-UDS, changing to, 2-2
encrypted, 2-4
FDT$, 2-2
in COBOL PCIOS format, 3-1
input, preparing, 3-1
preparing for input, 3-2
RDT$FILE, 2-2
SYSLIB*SORT, 4-1
SYSLIB*XRLOAD, 4-2
tape, 3-2
types, requirements, 3-1
UDS
availability, 2-2
exclusive assignment, 2-2
with read or write key, 2-4
XR$SELECT, 4-3, 5-3
XR$SORT, 4-2
XR$SORTOUT, 4-3, 5-3
FLOAT data type, for internal files, 3-8, 3-9, A-4
FORMAT item-position clause, 4-6
FTP files, 3-2

G

guidelines for tape files, 3-2

I

I/Os, reducing, 1-4
implicit storage areas, 2-4
improving load performance, 5-3
input files
approximate numeric data, 3-3
automatic assignment, 4-2
empty, C-1
exact numeric data, 3-3
EXTERNAL format, 3-6
in LOAD command, 4-5
INTERNAL format, 3-12, 3-13
loading into table, 3-6, 3-13
preparing, 1-5, 3-1
text requirements, 3-2
installation, 2-1
INTEGER data type, for internal files, 3-8, A-2
Integrated Recovery Utility (IRU)
DOWN command, 2-4
DOWN IMMEDIATE command, 2-6
recovery against incomplete updates, 2-5
recovery considerations
general, 2-5
long, 2-5
medium, 2-5
short, 2-5
STATIC DUMP command, 2-5, 2-6
integrated recovery, nonuse of, 2-5
internal files
ASCII COBOL data types, 3-8
description of, 3-7
input file, 3-12, 3-13
loading, 3-13
null indicators, 3-11
preparing, 3-2
requirements, 3-7
UCS COBOL data types, A-1

J

JETLOAD
  as XRLOAD storage manager, 1-3
  completion messages, 5-2
  function of, 1-5
  location, 4-2
  output, 5-2
  XR$SORTOUT file, use of, 4-3

K

kanji data, 3-4
key compression, bypassing, 4-3

L

LOAD command
  example, 4-9
  item-position field
    format, 4-7
    parameters, 4-8
  NULLIF clause
    format, 4-8
    parameters, 4-9
  parameters, 4-4
  syntax, 4-4
LOAD FACTOR parameter, 4-6
loading data
  improving performance, 5-3
  messages returned, 5-2
  speed, 1-3, 1-4
  XRLOAD process, 5-1

M

mass storage requirement, 2-1
multifile relations, 2-4
multiple vs. single LOAD command, 1-3

N

NCHARACTER data type
  for internal files, 3-8, A-1
  preparing, 3-3
null indicators
  internal files, 3-7, 3-11
  user files, 3-13
NULLIF clause
  format, 4-8
  parameters, 4-9
NUMERIC data type, for internal files, 3-8, A-3, A-4
numeric data, rules
  general, 3-3

O

offline loading, advantages, 1-2
options, XRLOAD call, 4-3
output
  JETLOAD, 5-2
  messages, 5-2

P

performance, improving, 5-3
preparing input files, 3-1
primary keys
  multifile relations, 2-6
  restrictions, 2-3
  sequence in internal files, 3-7
process of loading, 5-1
PROCESS STORAGE AREA command, UREP, 2-4

R

RDMUTL
  compared to XRLOAD, 1-3
  transition to XRLOAD, 1-4
RDT$FILE file
  availability, 2-2
  contents, 5-1
  DOMAIN attribute, changing, 2-2
read key, storage area with, 2-4
REAL data type, for internal files, 3-8, 3-9, A-4
recovery
  considerations, 2-5
  from incomplete load, 2-6
  point, 1-4
  procedure, 2-6
Relational Data Management System (RDMS)
  column definitions compared to ASCII COBOL data types, INTERNAL format, 3-8

column definitions compared to UCS COBOL data types, INTERNAL format, A-1
database, reorganizing, 1-4
datetime data EXTERNAL format representations, 3-5
extended format table definition, 3-6, 3-14
internal format table definition, 3-12
software level required, 2-1
table creation requirement, 2-3
UNLOAD statement
data, using for long recovery, 2-6
disk files, using as input, 3-1
input from, 1-4
unloaded internal input files, 3-7
reorganizing database, 1-4
runstream generated by XRLOAD, B-2, C-2
runstreams
execution result of XRLOAD, B-3, C-3
generated by XRLOAD, B-2, C-2
table definition, B-1
XRLOAD execution, B-2, C-2

S

schema
name of in JETLOAD output, 5-2
owned, DECIMAL data type definition, 3-8, A-2
unowned
DECIMAL data type definition, 3-8, A-2
SDF files, creating for input, 3-7
secondary indexes
CORE parameter in SORT processor, 4-6
database with many, 1-3
if numerous, effectiveness of XRLOAD, 1-3
restrictions, 2-3
sorting, 1-3
single vs. multiple LOAD commands, 1-3
site requirements, 2-1
SMALLINT data type, for internal files, 3-8, A-3
snap-ids on error messages, E-2
SORT processor
CORE parameter, 4-6
location, 4-2
performance, 4-2, 5-3
scratch file allocation, 4-2
use with XRLOAD, 1-3
work area size, 4-6
XR$SELECT/XR$SORTOUT file sizes, 5-3
XR$SORTOUT file, 4-3
sorting input records, 3-2
storage areas
accessed by multiple threads, 2-6
association with multiple tables, 2-4
CHECKSUM YES ignored, 2-4
conditions for, 2-4
corrupted, disabling, 2-6
creation, 2-4
deactivating, 2-4
FILE-TYPE EXEC, 2-4
implicit, 2-4
initializing, 2-4, 2-6
internal, 2-2
name of in JETLOAD output, 5-2
processing, deleting, and installing, 2-6
read/write key on, 2-4
registration, 2-4
reinitializing, 2-6
with table's primary key, 2-6
SYSLIB*SORT file, 4-1
SYSLIB*XRLOAD file, 4-2

T

table definition, B-1
tables
availability, 2-3
base vs. views, 2-3
conditions on before loading, 2-3
creation of, 2-3
definition, 3-6, 3-12, 3-14
foreign key constraints, 2-3
INSERT privileges on, 2-3
loading input file into, 3-6, 3-13, 4-9
multifile relations, 2-4, 2-6
overwriting of data in, 2-4
referential constraints, 2-3
storage areas
association with multiple tables, 2-4
CHECKSUM YES ignored, 2-4
FILE-TYPE EXEC, 2-4
reinitializing, 2-6
versions, 2-6
with NULL NOT ALLOWED columns, 3-11
tape files, guidelines, 3-2
tildes in input files, 3-11, 3-13
TIME data type, for internal files, 3-8, 3-10, A-5

TIMESTAMP data type, for internal files, 3-8, 3-10, 3-11, A-6

U

UDS cache hit ratio, 1-4
UDS files
  availability, 2-2
  DOMAIN UDS attribute, 2-2
  DOMAIN USER-UDS attribute, 2-2
Unisys Repository Manager (UREP)
  DD processor commands, 2-4
  PROCESS STORAGE AREA...DELETE command, 2-6
  PROCESS STORAGE AREA...INSTALL command, 2-6
UNLOAD statement, RDMS
  data, using for long recovery, 2-6
  disk files, using as input, 3-1
  input from, 1-4
user files
  introduction, 3-13
  null indicators, 3-13

V

views, restriction on loading into, 2-3

W

write key, storage area with, 2-4

X

XPC cache hit ratio, 1-4
XR$SELECT file
  function of, 4-3
  optimizing size of, 5-3
XR$SORTOUT file
  function of, 4-3
  optimizing size of, 5-3

© 2009 Unisys Corporation.
All rights reserved.

7847 4889–007